SKROLLI 2016.1E

SKROLLI

2016.1E
SKROLLI.FI

Computer culture magazine – International Edition

WHY DEMOS SUCK

EXCLUSIVE PLAYABLE DEMO: ILLUMINATUS

Build a medieval computer

Dumpster diving for hardware

Setting up a hackerspace

Cold War computers

Ville-Matias Heikkilä
editor-in-chief

Slow and steady

Skrolli is a paper magazine about computers, born in a world where paper media was already dying out because of computers. Contradictory and utterly insane, many would think.

To most people, computers are like social media feeds – short-lived, ever-changing and historically shallow. Any exceptions to the rule can be easily put under the comfortable umbrella of "retro". This is also the view endorsed by much of the mainstream, consumption-oriented computer journalism.

We at Skrolli see things differently. To us, there is no "retro". Computing culture, like any other culture, involves traditions, historical strands, ideas and material objects. They can always be used as a basis for something new – regardless of whether they are considered obsolete or not. And there is a lot in computing culture that is worth putting on paper and storing in the attic for decades. We are slow and proud of it!

But what is computing culture in a world where computers engulf every aspect of people's lives? We focus on where computing is irreducible: if you only casually chat or play Go with another person over the Internet, it is not really computer culture, because you can do it without a computer as well. Doing it with an AI, however, would be closer to the core, and the history and design of such AIs would be very close to what we are about.

Skrolli is a voluntary project run by a group of computer enthusiasts from Finland. At present, the project covers its running expenses, but none of us can yet do this for their main job. This is mainly due to the smallness of the audience that understands Finnish, and a large reason for the existence of this international issue is our desire to turn the hobby into a profession.

We have already published thirteen issues in our native language, and the articles in this issue are mostly translations of what was already available. You can probably notice our background in the emphases and some other peculiarities in the articles. Future issues are likely to have international contributions – maybe even from you – and, thus, more diversity as well.

Nevertheless, we picked some of our best. We gave the backers a translated list of articles and asked what they would like to see, and included some of our favorites in the mix. The variety of articles is somewhat similar to a typical Finnish issue – including hacking, programming, gaming, culture, history, hardware and a little bit of weirdness.

We hope you enjoy the first international issue of Skrolli! We also hope you spread the word about it and encourage your friends to buy it – because our future depends on it!

P.S. We promised our backers something extra, since our Indiegogo campaign was 127% funded. Please find a virtual cover disk with a playable demo on the cover!

Skrolli
Computer culture magazine

Contact: toimitus@skrolli.fi
Ircnet: #skrolli
skrolli.fi

Editor-in-Chief: Ville-Matias Heikkilä
Managing Editor: Annika Piiroinen
Art Director: Nasu Viljanmaa
Image Editor: Laura Pesola
Advertising Sales: Jari Jaanto
Finances: Anssi Kolehmainen

Editors: Mikko Heinonen, Päivi Julin, Jukka O. Kauppinen, Ronja Koistinen, Toni Kuokkanen, Teemu Likonen, Mitol Meerna, Manu Pärssinen, Janne Sirén, Suvi Sivulainen

Contributors to this issue: Albert Laine, Tapio Lehtimäki, Juuso Metsävuori, Andrew Gryf Paterson, Laura Pesola, Visa-Valtteri Pimiä, Ville Ranki, Oona Räisänen, Santeri Tani, Mikko Torvinen, Kalle Viiri

Translation: Mikko Heinonen
Proofreading: Michael Harlan Lyman
Publisher: Skrolli ry
Printed by: Hämeen Kirjapaino, Tampere,
ISSN 2323-8992 (print)
ISSN 2323-900X (online)

Cover image:
Tomi Väisänen

Pico-8

Fascinating fantasy console

When the existing machines no longer inspire you, it is time to switch to imaginary ones.

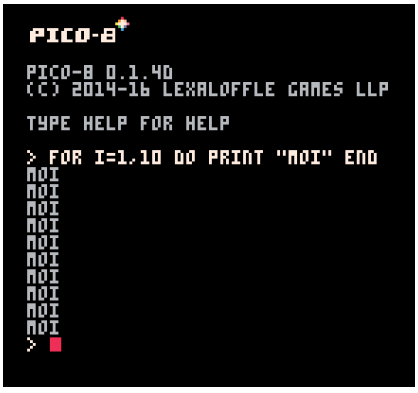
Story by Visa-Valtteri Pimiä, Ville-Matias Heikkilä

Images by Laura Pesola, Ville-Matias Heikkilä

During their careers, many computer hobbyists have tried countless device and software platforms. Some have even surveyed them intentionally in order to play their games or use them to create art. This sort of exploration also creates an understanding of the features that make a platform comfortable or interesting to work with. The fantasy console Pico-8 is one idea of an interesting platform.

The Pico-8 is probably best described as an emulator for a system that does not exist. Its overall spirit is very 8-bit and you might envision it as a handheld console like the Game Boy Color. The screen offers a resolution of 128 × 128 pixels at 16 colours and there are four chiptune channels for sound.

However, this is not merely an exercise in alternative history; the design of the Pico had very different starting points than devices that might seem similar on the surface. Instead of trying to make the most out of a limited amount of logic, it offers a small set of building blocks that are as fun to play with as possible. The developer hopes that, over time, the technical framework of the Pico-8 would give rise to a unique aesthetic that would be expressive in spite of its minimalism.



After start-up, the Pico-8 enters a command line console that can be used to input Lua commands.

First glance

The Pico-8 appears slightly schizophrenic. After start-up, it enters a mode that is more reminiscent of a home computer with a keyboard and mouse than a game console. The command interpreter allows for loading software and typing in *print* commands, for example. By pressing Esc,

Game cartridge. In theory, you could scan this picture and execute the binary on the Pico-8.


The sprite editor is used for sprites as well as larger pixel art.

you can enter an editor that has dedicated sections for code, graphics and sound. However, only the internal applications can access the keyboard, mouse or all parts of the file system – to external applications, the Pico is a game console with ROM cartridges and a two-button pad controller.

Pico speaks Lua, a limited but fairly expressive language that was originally designed for game scripting. In other words, the Pico virtual machine does not emulate any processor in itself – not even one that executes bytecode. Everything is written in Lua, and this is the lowest level a programmer can access.

There are two main formats for distributing games and other software. The cartridges, or carts, are PNG pictures that have the appearance of a physical cartridge and sticker, but whose lower bits store the program code and graphics and sound data like a watermark. Software exported to HTML5 format can be run on modern Web browsers without the actual Pico-8 software.

Bounds and limitations

The most visible technical feature of the Pico is the 128 × 128 pixel screen with a fixed 16-colour palette. The palette has a fairly personal and easily identifiable choice of colours, and its designer has clearly had more of an eye for colour than the average engineer.

Although Pico games typically use background maps consisting of 8 × 8 pixel blocks, and 8 × 8 pixel sprites on top of them, this is not a limitation. The graphics mode is a pure pixel buffer that can be used to draw anything – and the machine is also fast enough to run 1990s demo effects smoothly. However, the platform encourages the use of 8 × 8 pixel blocks by offering functions for drawing maps and sprites that are faster than using your own code to do the same pixel by pixel.

The cart can hold 15,360 bytes of compressed code. The maximum length in the editor is 65,536 characters or 8,192 tokens in tokenised form. These limits are not easily met – even many of the best games are clearly below these figures. On the other hand, the existence of this limit encourages simplicity and linearity, as there is no room for enormous game engines and multi-layered abstractions.

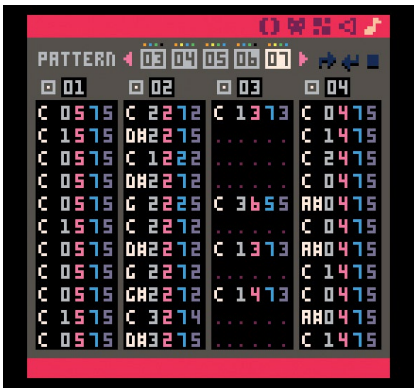
The cart has 12,544 bytes reserved for graphics and 4,608 bytes for sound. Of course, these data areas can be used for other purposes – the memory handling commands allow it to be accessed at the byte level. Upon program start-up, the contents of the cartridge's data side are copied into user RAM where the program can modify it, if necessary. User RAM also includes slightly under 7 kilobytes of space reserved for the user and 8 kilobytes of video memory.

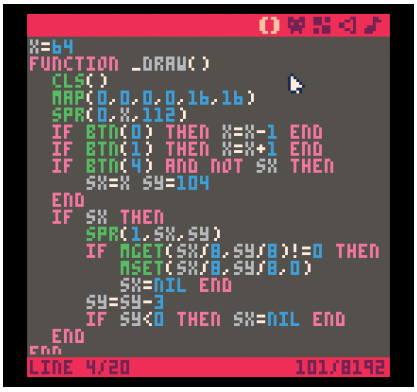
The graphics data consists of 8 × 8 pixel sprites where the entire colour palette can be used freely. The maximum number of sprites is 256 and the map consists of 128 × 32 sprites. It is possible to double the size of the map by settling for 128 sprites.

On the sound side, the equivalent of a "sprite" is a sound effect (sfx) that consists of 32 note locations. Each note location contains the note and the

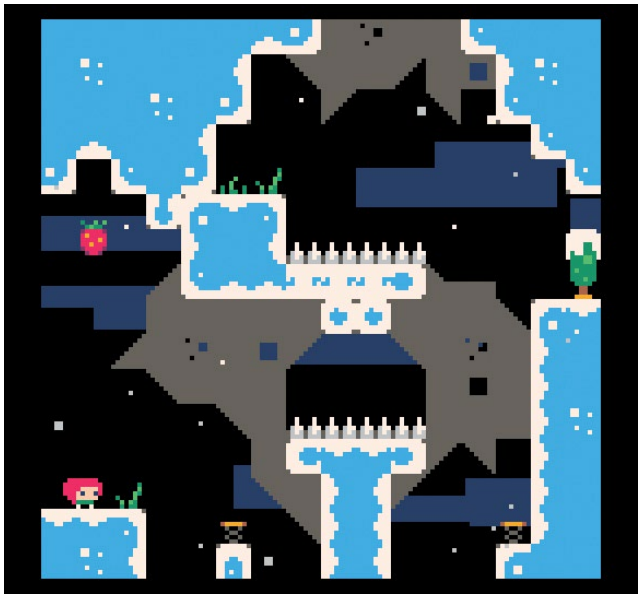

The map editor.


This mode of the sound effects editor is better suited for music. Actual sound effects are usually drawn as curves.


The music editor combines patterns into songs similarly to a tracker.

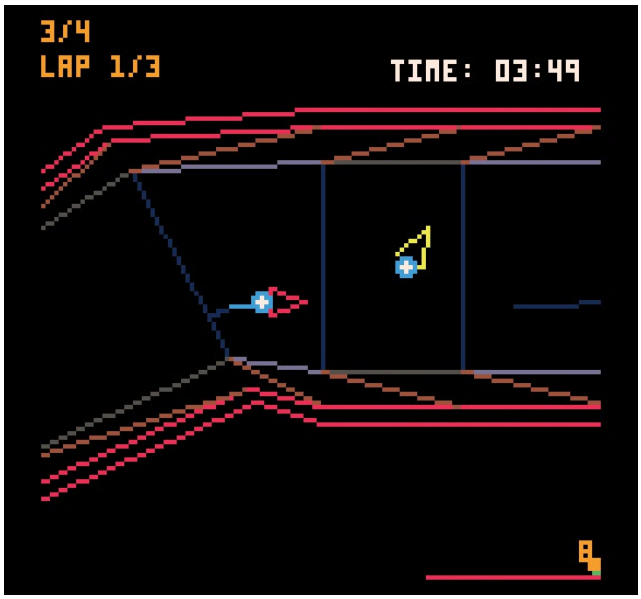


If the code editor seems too limited, you can always load the .p8 files in an external text editor.



Celeste by Matt Thorson and Noel Berry is a platform game with a steep difficulty curve.

Ennuigi by Josh Millard is more of a minimalist art film than a game. It is a mellow, melancholic interpretation of the story of Super Mario Bros.



Picoracer2048 is a line vector based racing game that is, unfortunately, single player only.

The atmospheric Dusk Child by Sophie Houlden mixes together adventures, puzzles and platforming.

waveform, volume and effect; there are 8 different types of each of these. The playback speed can be altered; lower speeds are better suited for music than sound effects.

Similarly to tracker music, a song consists of patterns that define which sound effect is played on each of the four channels. There is space for 64 patterns, which can hold several songs when loops and pattern end flags are used.

While the graphics side allows everything to be built from individual pixels, the user cannot access the "registers" of the sound system. In theory, you could build a player routine by modifying the sound data in real time, but the limits of the virtual machine's timing might not allow this. However, you can easily create different experimental soundscapes by writing random data in the sound effect memory.

In addition to the program code, data RAM and cartridge ROM, the Pico offers 256 kilobytes of space for the Lua interpreter. This is a relatively large amount when compared to the Pico's other memory spaces, but it will fill up easily with large tables, for example. One element of a number table takes up eight bytes, half of which is taken up by the actual data. Each number consists of a 16-bit integer part and a 16-bit fraction, which means that bit arithmetic operations can be used to compress them.

When running low on space, Pico can also read data from other ROM cartridges and even write to them. Program code has a strict limit, however; it can only be executed from the original cartridge. Lua in itself includes the possibility to execute data as code, but it has been removed in the Pico-8. This means that those requiring more code space will need to build their own virtual machine.

The Pico-8 does not execute Lua code as quickly as the computer's processor allows. Execution times have been de-

Paniq from the group Duangle is one of the demosceners interested in the Pico.

Hyperspace by J-Fry demonstrates that the Pico-8 can also do smooth, textured 3D graphics. As a game, it is nothing special.

fined for the different functions. This speed limit will rarely lead to problems during the development of typical Pico software, but it standardises the limits of the platform and prevents spiralling hardware requirements. According to the authors, a first-generation Raspberry Pi is enough for running even the most demanding Pico software at full speed.

How to code on it

Lua is written in all capitals and is, therefore, reminiscent of BASIC. For example, many people will remember the BASIC version of the following infinite text printout loop:

The Pico-8 Zine is not as thick as Skrolli magazine, but it is also available as a print magazine as well as in PDF format.

```
::START::
PRINT "HELLO"
GOTO START
```

Instead of the Lua standard library, Pico offers a fairly limited selection of BASIC-type functions: drawing commands, a couple of sound commands, controller input functions and a few functions for memory handling, mathematics, bit arithmetic and string handling.

The basic drawing commands can be used to draw pixels, rectangles, lines, circles, text, sprites and background maps. The palette colours can be switched for the drawing commands and you can also make colours transparent in terms of the sprites and background graphics.

Drawing moving graphics is more reminiscent of a PC than the 8-bit home computers. The Pico has no "hardware sprites" or "hardware scrolling". Instead, the display is usually redrawn for each refresh: clear the screen, draw the background and then draw the necessary sprites.

There are two functions available for drawing sprites: *spr()* draws an individual 8 × 8 pixel sprite at the provided coordinates, whereas *sspr()* draws an arbitrary area from a sprite sheet at arbitrary scaling. The scaling function enables a number of tricks that are fairly costly on most classic hardware, such as Doom-type texture mapping.

The programmer may place drawing commands in an endless loop, but a more elegant solution is to define a function called *_draw()* and call it at every screen refresh, i.e. 30 times per second. The earlier example would appear as follows:

```
FUNCTION _DRAW()
  PRINT "HELLO"
END
```

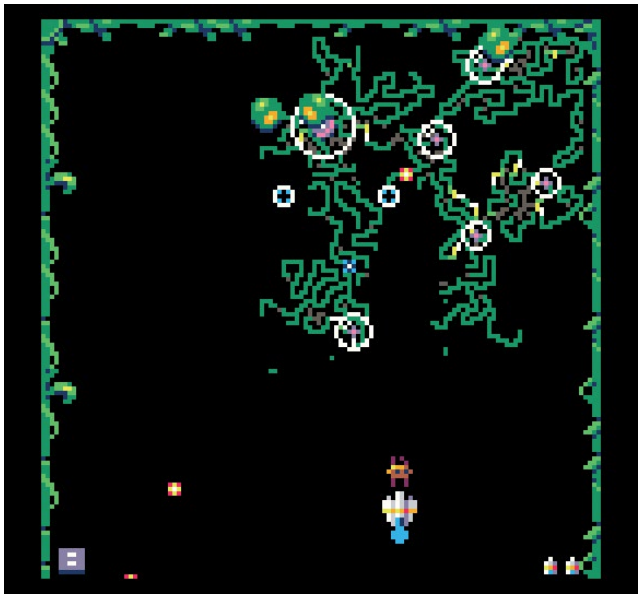
The game controller is read with the function *btn()* that accepts the button number as a parameter and returns whether the button is pressed. A program that moves sprite number 0 to the left and right might look like this:

```
X=64
FUNCTION _DRAW()
  CLS()
  SPR(0,X,112)
  IF BTN(0) THEN X=X-1 END
  IF BTN(1) THEN X=X+1 END
END
```

In order to display anything on the screen, you of course need to draw something for sprite 0 in the sprite editor.

Sometimes, *_draw()* will contain so many tasks that it cannot be run at every screen refresh. In this case, the programmer should move the updating of the game state to the *_update()* function that is – theoretically – called 30 times per second. Theoretically, because it is not a timer interrupt; if *_draw()* takes longer, it is called several times in a row.

Double-buffering is not a concern, as the changes to the video memory will only appear after *_draw()* has been



Hybris by Benjamin Soulén is a Japanese-style shoot'em up that takes place inside the human body.

Lemmtris by Movax13h combines two classics to create a fine puzzle game.

completed. On the other hand, the user RAM could only fit a full-screen double buffer if some of the sound effects were cleared out of the way.

The sound side offers the functions *sfx()* and *music()*. The former plays the sound effect given as a parameter on the first available sound channel, the latter starts playing music from the pattern number given as a parameter.

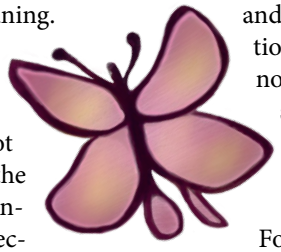
Makers of small 2D games need not concern themselves with the speed of the commands, but it will become an issue when testing the limits of the platform. The function *pset()* refreshes all the pixels on the screen approximately one and a half times during one screen refresh. Writing directly into video memory with the *poke()* function is about three times faster. However, using the *memcpy()* and *memset()* functions is up to ten times faster, and the background graphics drawing command *map()* is equally fast.

Blocks are fun to play with

Pico contains many features that are reminiscent of 1980s home computers. Instead of sticking to traditional or technological realism, it emphasises the fun of creating and block aesthetics. Those who are bothered by this should consider the fantasy nature of the platform: fantasy worlds do not always make a lot of sense, but they provide the setting for interesting events and fuel the imagination.

If we had to summarise the spirit of the Pico-8 into one word, it would probably be ”straightforward”. Some of this stems from the 8-bit computers and their BASIC: you can start writing your program immediately after ”powering on” and never need to think about OS requirements, APIs or different execution environments. Things are simple and tangible: specific bits in a specific memory location will always mean a block of a specific colour in a specific place on the screen.

The concept goes further than that, however: there is no counting of clock cycles and raster lines, no colour cell boundaries, no juggling back and forth with utilities and files. The limits of the platform prevent arduous and time-consuming fine-tuning. Since there are no adjustable palettes, sample systems or machine code instructions, you do not need to tune them. And the small number of pixels ensures that not even a perfectionist can spend very long with the anti-aliasing.



There are some technical challenges and brain puzzles on offer for those who desire them, but it is hard to imagine that success in the Pico-8 scene could ever require extreme attention to detail. Pico programs are easy and quick to write, and an author who is familiar with the basics of the platform can create a fairly polished game in a single evening.

At the time of writing this, the Pico is still an alpha version, and this is apparent in some parts of the development environment, at least. The code editor will not automatically skip to the line that contains an error and the keyboard cannot be used for drawing pixels. The software does not assist the user sufficiently; instead, the user needs to read separate documents and discover that the Esc key opens the editor, for example. The map editor can be initially frustrating, since it does not in any way indicate that the zero sprite is always empty in terms of the map.

Despite a few problems, we can recommend the Pico even to beginning programmers – at least those who are attracted to 8-bit block aesthetics and not afraid to read instructions from text files. Lua has no major drawbacks, and the simplicity of Pico's interface encourages doing things yourself instead of looking for ready-made solutions. For the more experienced, Pico offers an easy and fun pastime, and the results are way above simple doodling with watercolours.

Community and creativity

The development of the Pico-8 was crowdfunded, which ensured a large group of enthusiastic fans already at the time of publication. At the time of writing, the forum of the developer, Lexaloffle Games, has over 250 cartridges with games and other software. A substantial portion of these are platformers and other traditional 2D ac-

The Pico-8 demo Orbys by POD was succesful at Tokyo Demofest 2016.

Star Beast offers Wolfenstein-style perspective graphics.

tion games, but you can occasionally encounter some more experimental ones. There are even a few demoscene productions for the Pico.

The forum is the central community for developing and publishing on the Pico, and for the time being, at least, it has a very enthusiastic, warm and encouraging atmosphere. Beginners are also welcomed in a friendly and helpful manner. The Pico development team often participates in the discussion, which also makes the forum the best place to ask about the Pico's technical details.

Pico-8 also has its own fanzine, the Pico-8 Fanzine, which also receives input from Lexaloffle's main developer "zep". The creator of the indie platformer adventure VVVVVV, Terry Cavanagh, is one of the more famous writers. The fanzine is written by fans for fans, and three issues have been published at the time of writing. The zine contains instructions for making games, interviews, reviews and different technical articles for Pico-8 programming – and fan art, of course.

By now, many of our readers will surely be convinced that a software toy with such a warm-hearted spirit and an open development culture must be free and possibly even open source. This is not the case, however. The Pico-8 is a commercial product that currently costs around $20 to download. The binaries are available for the three most important x86 operating systems: Windows, Mac OS and Linux. Many have requested a physical Pico-8 console, but for the moment, it can only be implemented by using the runtime environment that can run the program binary.

Of course, the Pico is a simple platform, which makes it easy to reverse engineer, and the Lua interpreter it uses is already open source. However, the project is currently surrounded by such an aura of sympathy that not many hackers would have the audacity to produce an open and free Pico variant. Instead, we can hope that Lexaloffle itself will release the source code for the Pico when the paying customers start losing interest. This would make the Pico an interesting niche option for game development, and perhaps even education, after the community wanes and active development ceases.

The future of fantasy platforms

Fantasy consoles like the Pico-8 are a relatively new phenomenon. Although the Chip-8 virtual machine in 1970s hobbyist microcomputers can be considered its predecessor, and many educational software suites use simplified machines, the Pico is made exceptional by its "creativity first" approach. As such, its only predecessor is Lexaloffle's earlier Voxatron fantasy console.

It is quite possible that even more small, easily approachable fantasy platforms will start to appear in the wake of the Pico. And the motive does not need to be related to competition or contrast – curiosity towards other options will suffice.

The Pico's features encourage fun, cute and nostalgic creations, but a different selection of features could create an entirely different spirit and aesthetic. For example, it would be fun to see an "evil twin" of the Pico that emphasises the gloomy and rough corners of the universe of opportunities. Of course, this is a well-known phenomenon from the world of historical computing platforms, but a subculture that creates experimental fantasy platforms might offer interesting laboratory conditions for studying it.

Regardless of the future, the Pico-8 remains an interesting development environment that offers a more casual alternative for the classic home computers and consoles. It combines the inspiring 8-bit limitations with a modern design philosophy that emphasises ease of use and simplicity. Those who are even the least bit interested in coding and fascinated by large square pixels will find years of entertainment in the Pico.

Doomsday machines

Would you trust the fate of the world to a barometer?

It might be easy to forget this in a rush of nostalgia, but the 1980s were not only about moon boots, MacGyver and New Order. The threat of nuclear war between the United States and the Soviet Union was very real, in particular during the earlier part of the decade. And everyone knew what this would mean: the end of all life as we know it. This thought was not particularly comforting to 1980s children, and we would have slept even worse had we known how close to global nuclear war the world actually came.

Story by Mikko Heinonen

Images by Manu Pärssinen, Wikimedia Commons

Paradoxically, both the U.S. and the USSR were simultaneously afraid of each other and convinced that their counterparty would never think that they were going to be the first ones to press The Button. NATO's Able Archer 83 military exercise was a culmination of this absurdity. The allies were training for a scenario where nuclear war had started, which led Soviet espionage to believe that they were using the exercise as a cover-up to start a war. The agents were only asked to report their findings, not their conclusions – and chaos ensued since each one of them only saw a small part of the big picture. The situation was only defused when the exercise ended.

War on the big screen

The tension between the superpowers naturally left its mark on popular culture. There were dark doomsday prophecies such as Testament, Threads and The Day After, but nuclear war was a mainstay in movies of all kinds. One of the more famous ones is 1983's Wargames, where a young hacker called David (Matthew Broderick) accidentally calls a computer that supervises nuclear weapons while searching for the latest games. Even though the film does cut some corners in traditional Hollywood fashion, it contains realistic ways of discovering passwords, for example, and many have cited it as an influence for picking up computers as a hobby.

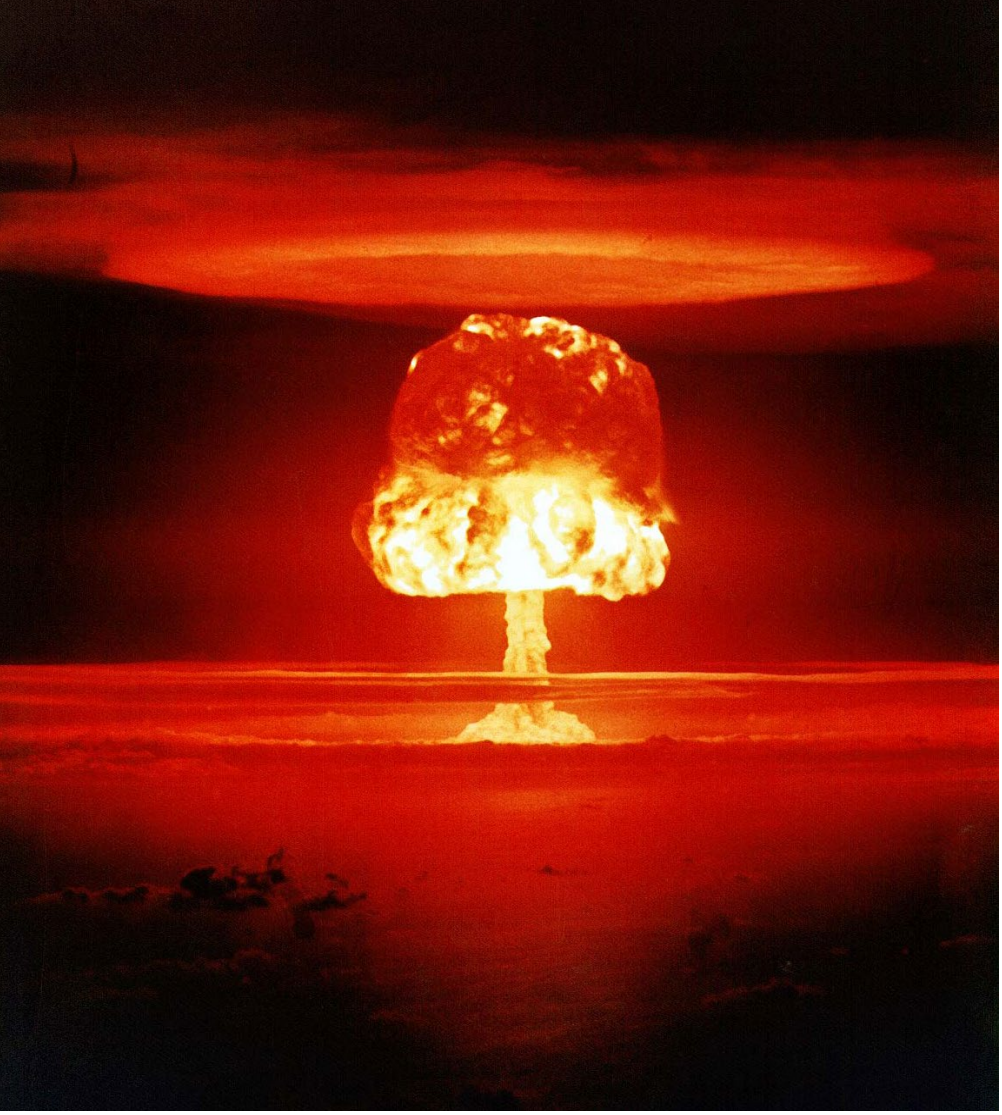
One of the main characters is an AI known as WOPR (War Operation Plan Response), developed by the fictional Professor Falken (John Wood). It tries to simulate different types of conflicts from tic-tac-toe to World War III. WOPR is commissioned because the operators sitting in the silos are unwilling to launch their nuclear weapons despite receiving direct orders. David accidentally sets the AI to simulate nuclear war, but the military command believes it has really started. WOPR also tries to launch the missiles autonomously until it reaches the reassuringly optimistic conclusion that the only winning move is not to play. It also becomes apparent that a machine's judgment can fail even worse than that of a human's.

Life imitating art

When viewed from the future, it seems that Wargames has more similarities with real life than the writers maybe even realised. The film was loosely based on an incident that took place in the early 1980s, during which the U.S. Air Force had already scrambled its nuclear bombers due to misleading information received from NORAD. The cause was not a hacker looking for computer games, but a computer stuck in "war simulation" mode. The on-duty personnel interpreted its messages as real. There had been at least two similar cases: one was due to human error, the other was caused by another malfunctioning computer. Even then, some branches of the military had not followed orders since they were sure that the alarm was false. By this time, the information had already reached everyone, which leads us to conclude that the risk of a nuclear holocaust was fairly low.

In September 1983, however, human logic helped to avoid the possibly severe consequences on the other side of the Atlantic. Officer Stanislav Petrov was sitting in a control room and monitoring signals from the Soviet missile warning system when he received a report of a Minuteman missile targeted towards the Soviet Union. Petrov reviewed the information and acknowledged it as a false alarm.

However, it did not take long before the system gave out another alarm; this time, there were four incoming missiles. Petrov reasoned that, in case of actual war, the United States would send hundreds of missiles instead of



only one. He switched off the alarm. This was a bold decision, but a correct one – it was later found that the device was malfunctioning. By disregarding the signal, Petrov might not have prevented the actual end of the world, but he certainly did his part in avoiding a conflict. He was shunned for his actions by the Soviets, but later rose to Internet fame.

The most interesting – or horrifying, depending on how you think about it – fact is that persistent rumours suggest a type of WOPR still exists. However, it is located in the Kremlin, not NORAD.

From my cold, dead hands

Sistema Perimetr, known in the West as Dead Hand, was part of the USSR's nuclear defence. Different sources offer inconsistent information concerning the actual characteristics, existence and operability of the system, and in particular the amount of automation involved, but they all have the same basic idea. When operators receive reasonably reliable information regarding a nuclear attack, they can transfer the decision-making authority to Perimetr. The system monitors changes in the ambient light level, radiation level and air pressure, and, according to some sources, also listens to signals from radio transmitters. It uses this information to determine whether it has been attacked. If this is the case, Perimetr will launch whatever is left of the nuclear arsenal.

The difference between Perimetr and the imaginary WOPR is that the former is designed purely for retaliation. At the same time, it is in fact designed to maintain peace, not to destroy humanity. When the final decision on launching a nuclear attack can be transferred to a system that is, theoretically, incapable of error, no human needs to press the button. This avoids the possibility of one or more human errors resulting in nuclear holocaust. Perimetr ensured that, if the Soviet Union were subjected to a surprise nuclear attack, the attacker would also be destroyed. In part, this would reduce the types of misunderstandings that occurred during Able Archer.

It is difficult to find reliable information on what Perimetr is actually like, which parts of it are operational and whether the system has ever been autonomous. A 2009 article by Wired magazine suggests that the system is continuously on stand-by and receives regular updates, but former Soviet generals say that it was never started. The reason behind the conflicting information is obvious: the Kremlin does not mind if the rest of the world believes that Russia possesses a unique nuclear retaliation system. In the West, Russia is considered a force to be reckoned with, even if it is no longer the "Empire of Evil" it once was. We may only know the truth after a long time, if even then.

But let us go back to Stanislav Petrov for a while. He acted as the interface between the machine and the physical environment and fulfilled his duties by dismissing the erroneous report. Who knows what would have happened if the seat had been occupied by Perimetr instead of Petrov. Perimetr's designers were undoubtedly aware of this, and it is hard to believe that anyone would have trusted an arsenal of the deadliest weapons to technology that is known to be unreliable. A more likely scenario is that the actual production system would have included its own Petrovs sitting in underground silos.

From master to servant

Despite the recent downturn in the relationship between Russia and the rest of the world, global nuclear war is, fortunately, still a fairly distant possibility. Nobody would seriously suggest using "infallible" computers to control nuclear missiles.

However, there are still more than enough nuclear weapons in the world, and computers are related to them more closely than ever. Since nuclear tests are no longer being performed, there is even more demand for modelling and calculation in weapon development. Control systems have also improved continuously.

When DEC introduced its 64-bit Alpha processor in the early 1990s, it had very modest software support. Evil rumours suggested that this was not a concern for Digital Equipment Corporation, since a cruise missile does not need a graphical user interface. While this was mostly unfounded speculation, the fact is that one of the largest Alpha based supercomputers was constructed at the turn of the millennium at Los Alamos laboratory – the very same one that developed the world's first nuclear weapons.

256-byte programs

These days, QR codes can be found everywhere. They are commonly used to store web addresses, but can just as well be used for any data – such as executable code.

Story by Ville-Matias Heikkilä

Skrolli magazine is preserving nine minuscule programs for future generations as QR codes – two games and seven demos. Although one code can store up to three kilobytes, the programs printed here are at most 256 bytes in length. Since this is a somewhat unorthodox use of QR codes, getting the software to run might prove somewhat challenging. We encourage you to experiment with them!

With the exception of one, all the programs are designed for MS-DOS. Read the QR code and store its raw text data in a .COM file. Then, use either a real PC or an emulator to run the file. The visually complex programs require more power than a typical DOS computer has – using the DOSBox emulator's turbo mode (Alt+F12) is recommended.

One of the demos is a piece of JavaScript that may even run on some hardware with no modifications. It is nothing special and is mainly intended as a proof of concept. The amount of framework code makes JavaScript a less suitable platform for extremely small programs than MS-DOS.

4is256 (Řrřola, 2007) – A Tetris clone. Control with Shift, Ctrl and Alt.

Boulder Dash in 256 bytes (James David Chapman, 1995) – A rudimentary version of a classic game. Find the exit and beware of the falling rocks!

Bump is Possible (Downtown, 1999) – A rotating concrete torus.

Dírojed (Řrřola, 2007) – Psychedelic feedback in 32 bytes.

JavaScript test demo (Skrolli, 2013) – A simple canvas effect that our readers can improve on!

Puls (Řrřola, 2009) – Animated machinery built with raymarching.

Searchlight (Wamma, 2007) – A raycast catacomb with columns that cast shadows.

Sqwerz3 (Trimaje, 1996) – Rotation among multi-coloured squares.

Tube (3SC, 2001) – A freely rotating camera inside a spiral tunnel.

History of Soviet and Eastern Bloc computers

In order to make the most out of this voyage through the history of computing in this great nation that is no more, we recommend that you play the Soviet national anthem in the background. Full volume is preferable.

Story by Jari Jaanto, Ville-Matias Heikkilä Images by Sächsische Landesbibliothek, Andrei Kulikov, Wikimedia Commons (Panther, Konstantin Lanzet, Pereslavskaja pedelja, NZeemin, Arseni Gordin, SysCat, Andrew Butko)

ES-1035, an ES EVM series computer, in use in East Germany.

The Soviet Union pioneered many great advances in the fields of science and technology. It was the first nation to send a satellite into orbit, the first nation to send a human into space, and the first nation to set up a long-lasting space station. The USSR put heavy emphasis on scientific and technological research, which made it a commendable adversary to the Western world.

It entered the development of computer technology at the turn of the 1950s. The first programmable electronic computer in Continental Europe was the MESM (*Малая Электронно-Счетная Машина)*, built during 1948–1950 and commissioned in 1950. The military industry was already using electronic calculator applications and analogue computers, but the age of the digital computer started with MESM. It consisted of 6,000 electron tubes and consumed 24 kW of electricity. It could perform approximately 50 calculations per second and it was used in top-secret nuclear weapons projects.

Setun

There were many innovations during the early stages of Eastern Bloc computing. One of the most eccentric machines was the Setun (Сетунь) that was built for research use at Moscow State University in 1958. Instead of the regular binary system, it used what is known as a balanced ternary system. This system includes minus one in addition to zero and one.

The Setun was a technological success. This was attributed in particular to the ternary system. It proved to be

Setun.

-6	1010	-+0
-5	1011	-++
-4	1100	0--
-3	1101	0-0
-2	1110	0-+
-1	1111	00-
0	0000	000
+1	0001	00+
+2	0010	0+-
+3	0011	0+0
+4	0100	0++
+5	0101	+--
+6	0110	+-0

Numbers from -6 to +6 in decimal, binary and balanced ternary.

Ural-1, high-end Soviet computing from the 1950s.

reliable and stable at different temperatures and supply voltages, and it was simple to construct and operate. Approximately 50 units were built by 1965. Although the Setun also gathered interest in the Western world, the decision-makers considered it too eccentric for the planned economy and the project was suspended. The Setun at Moscow State University was replaced with an equally powerful binary computer, but the operating costs of the new system multiplied.

In 1970, the Setun received a successor called Setun-70. It was programmed using the DSSP language that was reminiscent of Forth. The Setun and Setun-70 were the world's only ternary computers.

Argon-16

In the 1960s, development was divided into civilian and military branches, and space computing was a part of the latter. Fighter jets, spaceships and surveillance posts needed light, failure-tolerant control computers, and these could be found in the Argon range. The most famous of the series is the Argon-16 (Аргон-16). It was used in all the Soyuz and Progress spaceships and for 37 years at the Saljut, Almaz and Mir space stations. The machine never failed, which makes it the most reliable computer model used in space. It is also the computer with the longest service record in space.

The Argon-16 went into production in 1974. All of its components, including memory, had triple redundancies that all worked at the same time. The main design criteria for the computer were reliability and real-time cooperation with the other equipment of the spaceship.

Since 2010, Soyuz spacecraft have used a new control computer called TsVM-101 (ЦВМ-101) in place of the Argon.

Copying Western computers

In 1966, the financial planners suggested developing a series of computers that would include models with different hardware designs but compatible software. This created the unified system for electronic computers, ES EVM (ЕС ЭВМ), which went into production in 1972. Computers in this series were manufactured in the Soviet Union as well as in the other socialist countries, and they even outlasted the collapse of the Soviet Union itself, since the final models were produced in 1998.

The unified system allowed compatibility with Western computers for the first time. Although there were competent Soviet hardware designs available, the system was built on the American IBM S/360. IBM did not mind this and entered into cooperation negotiations with the Soviet Union; however, they ended in 1979 due to the United States' economic sanctions.


This Soyuz flight would not have succeeded without the Argon control computer.

Other computer series based on

Elektronika 60M.

BK-0011M.

Western technology were the SM EVM (PDP-11 and VAX) and, in the 1980s, the ES PEVM (IBM PC). In addition to these central product lines, other devices, such as the Apple II, Oric and ZX Spectrum, were also cloned. Cloning the Intel 8080 processor made it possible to build CP/M compatible computers. The Cray-1 supercomputer was also successfully cloned in the late 1980s.

Microcomputers

When Intel introduced the world's first microprocessor in 1971, the Soviet Union was not yet far behind: The K145IP1 processor was developed in 1973 and the first pocket calculators using it were introduced to the market in 1974. Although the politburo emphasised the production of pocket calculators, it was not interested in personal computers which started gaining popularity in the West in the late 1970s. The country was going through financial difficulty and could not even meet the growing demand for computers in its research and production facilities. Starting in 1980, the neighbouring country Bulgaria produced Apple II compatible Pravetz computers for educational use, but production figures were initially low. Since CoCom, led by the United States, had imposed an export ban on high technology, the only options available to regular Soviet citizens in the late 1980s were DIY and smuggling.

In 1982, the hobbyist magazine Radio (Радио) published assembly instructions for a computer known as Mikro-80 (Микро-80). The computer used Soviet K580 series processors that were copies of the Intel 8080. It had 64 kilobytes of memory and the only display mode was a text mode with 64×32 characters. The computer had over 200 parts, which is why only a few were ever built by hobbyists. Sourcing parts was difficult and could only realistically take place through the black market. A simplified version of the Mikro-80, the Radio 86RK (Радио 86РК), was released in 1986. The computer had been reduced to 29 components. The processor and text mode were the same as before, but memory had been cut to 16–32 kilobytes. The machine gained some popularity among hobbyist circles and inspired several variations. It also saw several add-ons, such as an accessory that could display graphics.

In the same year as when the Mikro-80 was released, the Soviet Ministry of Radio Industry also introduced an Apple II clone known as the Agat-4. Whereas the Bulgarian Apple clones used their own clone of the 6502 (the CM630), the Soviet models replaced it with an interesting simulation arrangement. The production batch in 1984 was very small and only the later versions, Agat-7 and Agat-9, spread into mass production. Apple II compatibility improved with the new models. The Agat was popular in education, but hobbyists never received it in large quantities.

Western computers were not as difficult to purchase in all Eastern Bloc countries as they were in the Soviet Union. Black market Commodore 64s were fairly popular in Poland, Yugoslavia and East Germany but completely unheard of in the Soviet Union. The only capitalist home computer officially sold to the Soviet Union appears to be the MSX which Yamaha sold for use in teaching.

Elektronika

In the West, the PDP-11 instruction set was solely used for machines the size of refrigerators; in the Soviet Union, however, it was a very popular choice and used in smaller devices and embedded systems, right up to satellites and graphing calculators. The small PDP-11 compatibles used the same Elektronika (Электроника) product name as the pocket calculators. The Elektronika models 60, 85 and DVK were designed for terminal and workstation use. The model 60 is known as the computer that was used to write the original Tetris. The workstation range was also used as the basis for the first official Soviet home computer, the Elektronika BK (БК). The BK was mainly intended for use in schools, but it was the first computer that Soviet citizens could officially purchase for the home.

The first BK series computer, the BK-0010, went into production in 1985. Its processor, the K1801VM1, was 16-bit in accordance with the PDP-11 architecture and operated at

"Soviet Amiga" UKNC or MS-0511.

Pocket computer Elektronika MK-90.

The control unit of an ES-1052.

Nu, Pogodi! handheld game.

a clock frequency of 3 MHz. It had 32 kilobytes of RAM. There were two display modes: 512×256 with two colours and 256×256 with four colours. These modes were also used to simulate 64×25 and 32×25 pixel text modes. The machine shipped with the FOCAL programming language and a BASIC interpreter was optional.

The BK-0010 received an improved version, the BK-0010.01, which had a typewriter keyboard in place of the old numb keypad. It also had the BASIC interpreter in ROM. The BK-0010Š was released for schools, and it included a monitor and networking functionality. The BK-0011M, released in 1989, upped the clock frequency to 4 MHz and the RAM to 128 kilobytes. Graphics were also improved: previously, the only colours on offer had been black, red, green and blue, but now there was a choice of 15 other four-colour palettes.

In the 1990s, users of the BK-0011M fitted the machines with AY sound chips and controllers for disk drives and hard drives. BK demos started appearing with the rise of the Spectrum demoscene.

Other computers

The UKNC (УКНЦ) computer introduced in 1987 could be called the Soviet Amiga in terms of its appearance and inner workings. The UKNC had two PDP-11 compatible K1801VM2 processors, one of which acted as the graphics processing unit and auxiliary processor. The main processor had a clock frequency of 8 MHz, the machine had 192 kilobytes of RAM and the graphics mode offered 8 colours at a resolution of 640×288 pixels.

However, the "Soviet Amiga" is bypassed in terms of its multimedia features by the Vektor-06C (Вектор-06Ц), which was an 8-bit home computer developed in the same year. While other Soviet computers only used beep sounds, the Vektor had a dedicated clone of the Intel 8253 counter chip for sound. It offered three square wave channels. The total colour palette was an impressive 256 colours, of which 16 at a time could be used in the 256×256 pixel graphics mode. The bottleneck of the system was its main CPU, a clone of the 8080 that could not refresh the display quickly enough at a clock frequency of 3 MHz. The computer had 64 kilobytes of RAM, half of which was video RAM. The device was fairly popular and a lot of games were made for it.

While on the topic of Soviet information technology, we must not forget pocket calculators. Some of them were also sold in Finland through the Teboil chain of service stations. There were countless models of calculators, from simple standard models to programmable graphing calculators, the finest of which was the restrictively expensive Elektronika MK-90. The MK-90 had a 160×64 pixel liquid crystal display, a BASIC interpreter and a PDP-11 compatible processor. The clock frequency had to be kept low, which made the device operate slowly. The calculator shipped with an empty storage module and a ROM module that contained Tetris, a clone of Pac-Man and a chess game, among other things.

The socialist bloc also produced electronics solely designed for gaming. The Soviet system churned out handheld electronic games, most of which were direct copies of Nintendo's Game & Watch games, and coin-operated arcade games. Most arcade games had gigantic pixels and very modest tech-

These Bulgarian Apple clones appear to work fine in a school in Pereslavl in 1985, but is the heating working?

nology, but some were surprisingly advanced. The TIA-MC-1 hardware had 256 colours and sprites, among other things.

Out of the larger computers, we should mention the Elbrus series that started in the 1970s. It includes multiprocessor supercomputers that were used for space and nuclear research. The developer of these machines, Boris Babaian, has been called the Seymour Cray of the Soviet Union.

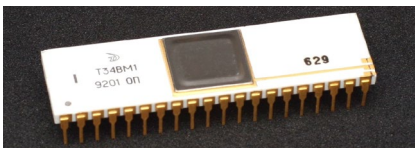
Cloned 8-bits

Although the Soviet Union and the Eastern Bloc did produce many original computers, the most popular home micros were copies of the British ZX Spectrum. Spectrum was the ideal machine for cloning. Its small size made it simple to smuggle across borders, and technological simplicity made it simple to reverse engineer and manufacture. The first Soviet Spectrum was built in 1985, and by the early 1990s, nearly all of the larger Soviet cities had Spectrum manufacturers. Some of the Spectrum clones became much more sophisticated than the machines they were based on.

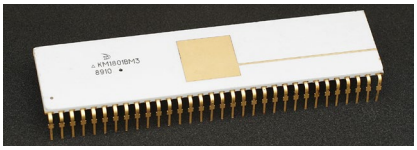
Although the Spectrum was by far the most popular 8-bit computer in the socialist countries, other 8-bits were also being cloned. The Agat and Pravetz, mentioned earlier, were Apple II clones, except for Pravetz 8D, which was an Oric clone. In Yugoslavia, a ZX81 clone called Galaksija was introduced already in 1983; it utilised Z80 processors imported from the West. NES consoles have been cloned in China since the late 1980s, and some of them ended up on the Soviet and Eastern European markets.

K1801BM1.

KP580BM80 (= Intel i8080).



T34BM1 (= Zilog Z80).



KM1801BM3 (KM1801WM3).

Hobbyist culture

It was typical of the Soviet computer industry that software production was omitted in the five-year plans. Hobbyists and professional users alike had to write their software from scratch and perform different hardware modifications. This created a strong maker culture and a black economy that can be compared to the uncommercial nature of Western hacker cultures. While software production was big business in the capitalist countries and copying mainly took place at universities and in hobbyist circles, the opposite was true in the socialist countries. Software piracy was the generally accepted norm and money only exchanged hands in the hobbyist circles and on the black market.

Games and software for the BK computer, in particular, typically contain a start screen that can be compared to the crack intros by Western pirate groups. This screen introduces the author or distributor of the software, advertises other games on sale and provides a phone number for further information. There was a large amount of unofficial software production for the BK. Hobbyists converted both Western software and software from other Soviet computers and produced a number of games.

The first demos from the socialist countries were made in Poland and Czechoslovakia for the Spectrum, from where the phenomenon migrated to the Soviet Union through the software piracy networks. During the initial stages, Eastern Bloc demo authors had no knowledge of the existence of the Western demoscene. They started writing demos purely from their own perspective. The Soviet hobbyists encountered the Western demoscene for the first time through Amiga demos, even though none of them had Amigas. Demos spread as copies of copies of VHS tapes, and Spectrum hobbyists used them for inspiration. A few demos contained the phrase ”Amiga rules!” even though none of the authors had ever seen a real Amiga.

The demoscenes of the East and West only discovered each other years after the collapse of the Soviet Union, in the latter half of the 1990s, when the ”ex-USSR” demo culture stabilised and the arranging of regular, larger demo events started. At this time, the main platform for Eastern Bloc demos became the Spectrum clone Pentagon 128 that is still the de facto standard for Russian Spectrum demos.

Summary

Soviet era home computers typically had a Western clone processor, but the rest of their technology was original. Depending on the time and manner of calculation, technology was 5–10 years behind Western hardware. Building computers yourself, which was typical of early 1970s microcomputer culture, was still common in the Soviet Union in the latter half of the 1980s. Software and peripherals were not sold and had to be built by the hobbyists themselves. The Soviet era had a strong DIY culture that still lives on!

- MESM: http://www.engadget.com/2011/12/26/mesm-soviet-computer-project-marks-60-years/
- Setun: http://en.wikipedia.org/wiki/Setun
- Moscow State University: http://hpc.msu.ru/?q=node/57
- Setun-70: http://ternary.3neko.ru/setun70.html
- Agat: http://en.wikipedia.org/wiki/Agat_computer

We upgraded your device, it's ruined!

Any love for an outdated android?

Story by Mikko Heinonen

I keep an Android tablet on my bedside table and use it to view online content before going to sleep. A Google Nexus 7 (2012) worked fine for this purpose for about two years.

All the software I needed worked nicely, until Google updated the device with Android 5.0. Upon first start-up, it was clear that the new OS alone was eating the resources of my humble tablet. Moving between applications was sluggish and video playback, in particular, slowed down to a crawl.

When I brought this up with an Android veteran, the response was clear: it's all your fault; you should have never upgraded. Owners of old devices should have stuck with the old version, which also uses up less of the resources. However, this is problematic because Android also has its share of security vulnerabilities, and I sometimes use the same tablet to book tickets while travelling, for example. I want to ensure that the security of the device is as up to date as possible – and, going forward, these fixes will only be available for the latest version.

Well, at least somebody cares

At the same time, I realise that I am fortunate. Google is interested in providing new software versions for its reference devices even after they are two years old. At home, we also have a newer, cheap Chinese tablet that has received exactly zero OS updates during the last year. From now on, it will need to rely on community-built custom ROMs.

Even larger manufacturers tend to abandon their devices early, however. A year ago, factory refurbished iPhone 4s were offered very cheaply and people everywhere rushed to get them. Not many of them considered the fact that their newly purchased phone had at least one serious vulnerability that Apple has no intention of ever fixing. The phone is five years old and thus considered an end-of-life product that receives no new OS versions.

A change is coming

The end of support for Windows XP was also a hot topic a while ago. The old workhorse was finally taken off life support after 13 years, and it is still dearly missed by many. There are still millions of active XP installations, many of which will, no doubt, cause all sorts of joy to their owners and network operators, among others.

Updating XP was expensive for Microsoft. Buying a cheap OEM licence in 2001 entitled you to a decade of free updates. The new versions were not radically different, which made XP too popular for it to be simply discontinued. The support for Windows 7 was also extended until 2020, since the transfer from XP to 7 started late.

This is clearly a problem that Microsoft aims to avoid with Windows 10. Offering the upgrade for free is designed to lure users towards the latest version and to relieve the historical burden. They are clearly borrowing pages from their opponent's playbook, since Apple has already been distributing free OS X updates for a longer time. The only difference is that you can only run OS X on Apple hardware (without resorting to dirty tricks, that is). Microsoft needs to offset the difference with advertising income.

Sour Apples

A few months ago, a relative of mine asked me to take a look at her old MacBook. It was maybe 6–7 years old, but still mechanically intact and sufficient for her purposes. Unfortunately, though, the integrated display adapter was so old that a new version of OS X would not run. This, in turn, prevented the installation of later software versions. The reality was that, in order to run modern software, she either needed to switch to an open source OS or head to the store.

Another friend of mine managed to upgrade the OS on her old Mac, but the end result was a complete loss of performance. After a few weeks, she grew tired of waiting and bought a new one. That is also what I did with my sluggish tablet.

Walking time bombs

Offering the latest update for free is a handy way for manufacturers to reduce their duties. There will be no need to fix ancient OS versions when everyone is offered the latest edition. After this, they only need to decide which hardware to make obsolete during each round.

Owners of old devices can appreciate the idea of a new operating system – until they realise that they are actually faced with an unpleasant choice: should they accept the vulnerabilities and other issues or slow their device to a crawl?

This decision is easy to make as long as the consumers can afford to – and can be bothered to – update their hardware every two years or so. However, this is not even nearly always the case, and even the old devices are usually sold or handed down to someone. The world is full of yesterday's hardware that receives no software updates. Usually, an average user cannot understand why they should replace a working PC or phone that still serves its purpose. Or why they should look into GNU/Linux at the latest when the hardware starts to show signs of ageing.

After all, it is true that Windows XP can still be used to view news on the Internet. A phone with an old version of Android or iOS will still work when making calls or posting on Facebook. This becomes a problem, however, when millions of these walking dead are connected to the Internet. The creation of a zombie army is only one major vulnerability away.

Abandonware
– the controversial software graveyard

What is abandonware? Who abandons software and why? What makes one person's trash another person's treasure?

Story by Mikko Heinonen, Kalle Viiri Image by Tapio Lehtimäki

Abandonware refers to commercial software whose copyright is not being actively enforced. In many cases, the software has not been commercially available for years. The publisher may have gone out of business entirely or switched the focus of its business, or the software may have been published for a device that is no longer commonly used. While the software is still technically subject to copyright, no-one is enforcing it.

Since the amount of available software is enormous and continuously growing, this graveyard has a constant supply of cadavers. Some of them will later rise from their graves, but many remain in limbo forever.

Players first

Games are the best-known and most popular form of abandonware. This is due to a number of reasons. The number of games being published is huge, which results in a higher rate of abandonment. Games are usually made with small budgets and the studios creating them are often short-lived. Furthermore, productivity software usually develops in a manner that makes older versions obsolete.

Games, however, behave differently. Their changes are commonly related to the technical implementation. Many old games have features that players keep coming back to even after several years. Players also commonly have a strong sense of nostalgia towards games from their childhood.

Like all other software, games have been copied and distributed for as long as they have existed. The concept of abandonware, however, can be seen to have been brought about by the emergence of the World Wide Web, since it allowed for offering entire, curated collections of abandoned games and their descriptions. Many games were already fairly old by the time WWW came around.

It is no wonder, then, that the Internet is full of sites that offer classic

games and other software for download. They all operate on more or less the same principle: the software is made available because the webmasters believe that the copyright holders have lost interest in it. Some sites play it safe and remove the download links when the software becomes available again. Copyright holders are also encouraged to request the removal of any downloads. While these disclaimers do not really hold water in legal terms, many of these sites have been allowed to operate for years with no major legal repercussions.

Grey areas

In a way, abandonware only exists until someone makes a decision on the fate of the software. If the copyright holder decides that they have no more interest in the software, it becomes either *freeware* or, in the best case, completely open source. In the latter case, it falls below one of the many open source licences.

id Software, for example, has released the source code for many of its games and game engines, while retaining the rights to the rest of the game data. Similar examples can be found from the field of productivity software, as well. Many other developers, such as Mr. Chip (Magnetic Fields) who were behind the successful Kikstart motorcycle game, have released the binaries of their old products on the condition that no profit is made from their sales. Others, such as Cinemaware, the developer of interactive movies, have been distributing emulated versions of their own games in order to create traffic for their website and, as a result, generate interest towards their new products.

But there are opposite examples, as well. Some publishers have made it a policy to keep a firm grip on their intellectual property. Nintendo, in particular, is known as a watchful guardian of its rights. This is, of course, related to the fact that Nintendo continues to offer its old titles for purchase to the owners of its latest gaming devices. Its lawyers have shut down several sites distributing the adventures of Mario and his pals for free. This has resulted in the addition of a specific ”non-Nintendo clause” on some abandonware sites. Naturally, this has done little to prevent the online distribution of these games, since new sites are born as soon as others are closed. Distribution has also shifted from WWW towards peer-to-peer networks that are more difficult to trace.

Rise from your grave and make money!

Not every game is forgotten forever. Different people have gone to great lengths to locate the publishers of classic games and ask for permission to publish them – or even to acquire the publication rights. Street Rod, a classic car tuning simulator, is now legally

available for free download thanks to the efforts of a private individual. The rights to the games of the ancient Intellivision game console are held by Intellivision Lives!, a company set up by hobbyists. They have even released a compilation disc that works on Windows. And there are countless similar examples.

GOG.com, previously known as Good Old Games, was established in 2008, and it is probably the best-organised reseller of old games. Its original business idea was to locate the copyright holders of old games, sign distribution agreements with them, and republish games on their own digital distribution channel after adapting them to the latest hardware. The development of the open source DOSbox emulator was highly beneficial to GOG in this respect. Later, they also expanded to newer games.

GOG.com offers over one hundred games that were released before 1995. This is a good selection of early DOS classics, and making the games available for only a few dollars has made them a feasible alternative to illegal downloads. Furthermore, since GOG publishes the games without any form of copy protection, it is completely possible to extract the original game files from the package and play the GOG version on a real DOS computer.

Commercial nostalgia trips are also available for other popular platforms, such as the Commodore 64 and Amiga. A company called Cloanto offers the commercial products Amiga Forever and C64 Forever, which include a licence for the ROM files of the original computer and a selection of games. Nevertheless, a nearly complete, carefully curated library of games for both systems has been readily available online for a number of years, and nobody seems to mind.

However, the field of abandonware also has its share of shadier entrepreneurs. Chinese online shops commonly offer devices that can emulate a variety of 8-bit and 16-bit game consoles. Very often, they also ”accidentally” contain commercial ROM files from these systems; at worst, this is even listed as a feature on the packaging. The original copyright holders were not consulted, of course.

Bio Menace (Apogee 1993, DOS).

Preserving culture

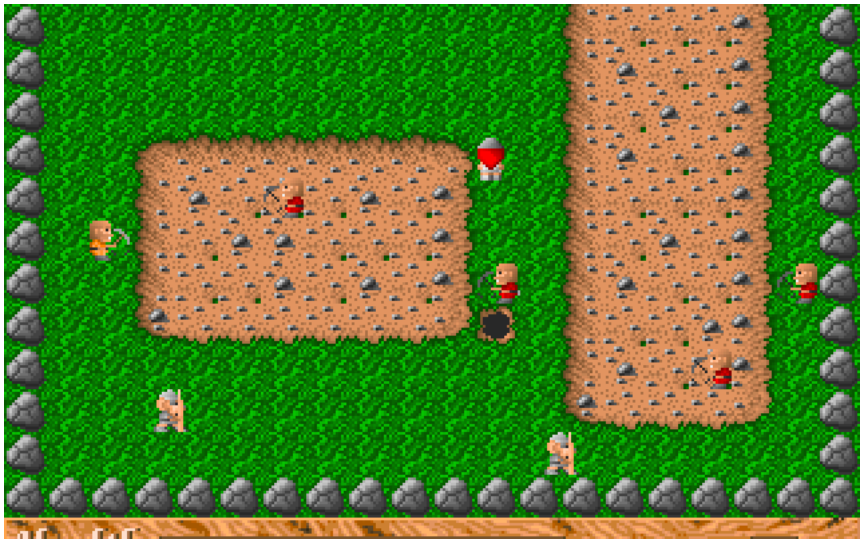
Sites distributing abandonware have reunited countless players with their childhood favourites, but they also serve a nobler purpose. The preservation of digital software is far from systematic, and even the original authors no longer have copies of some software. And since many publishers who are active today have admitted that their archives have been lost a long time ago, what can we expect from those who went out of business? Very often, research into the early stages of gaming history has utilised disk images and box scans found online. What was originally a form of piracy has become the preservation of cultural history.

Writing emulators, which are es-

sential for the preservation of history, would also be substantially more difficult without an enormous, readily available library of software. Game programmers, in particular, were notorious for utilising every available quirk in the hardware, and emulators that only rely on the official documentation will not operate correctly under all conditions. A large amount of different software is required in order to ensure the functionality.

The graveyard needs a rulebook

Abandonware is an unclear concept. At the moment, anyone can distribute any software they like and claim that it has been abandoned. In practice, the person downloading it is responsible for determining whether the site is authorised to distribute the files. There have even been cases of fraudulent licence terms. A version of DOS is in circulation that calls itself MS-DOS 7.1. It has been extracted from Windows 98 and the text of the GNU GPL has been appended to it. Microsoft has not released the source code for DOS 7.1 and does not appear to have any intention of doing so in the near future.



God of Thunder (Software Creations 1993, DOS).

In 2013, an initiative was launched on the Finnish Citizen's Initiatives service for limiting the duration of copyright for digital software unless the publisher requests an extension for it. The rationale for the initiative was similar to that used in abandonware: software that is over twenty years old is commercially feasible only in very rare cases. The initiative did not even come close to gathering the necessary number of signatures, and EU legislation as well as international copyright agreements would have made it nearly impossible to enact in any case.

The matter is too important to ignore, however. At the moment, years of software preservation can be lost in an instant if a copyright holder decides to shut the operation down after remaining passive for a long period of time. In practice, the preservation of digital culture currently relies on benevolent software pirates, publishers releasing their work to the public and vague concepts such as abandonware.

One Must Fall 2097 (Epic MegaGames 1994, DOS).

Internets before the Internet
– the rise and fall of modem BBSes

Finland has been connected to the Internet for over 25 years. However, computers were already talking to each other long before this, and it took years for the Internet to supersede the other networks.

Story by Ville-Matias Heikkilä Images by Manu Pärssinen, Ville-Matias Heikkilä, Wikimedia Commons

Machines talking on the phone

The construction of electronic data networks started in the 19th century, and one of them eventually reached nearly every home. This was, of course, the telephone network. Finns were enthusiastic about building networks from the start: By the 1890s, Helsinki had several telephones per one hundred residents, which was one of the highest ratios in the entire world.

The telephone network was also an attractive option for relaying other information in addition to speech. News agencies had to send a lot of information to each other quickly, and this was done by using teletypes, which were a type of remotely operated electronic typewriter. Teletypes had their own Telex network, but the telephone network reached further and was, in most cases, cheaper to use. You only needed a modulator/demodulator, or ”modem” for short, between the teletype and the phone line in order to convert the character data to different tones and back again. The first phone line modems were taken into use in the 1940s, which means that the technology for connecting computers to the telephone network pre-dated the computers themselves.

The first widespread computer network was SAGE (Semi-Automatic Ground Environment), commissioned by the United States military in 1958. It mainly relied on regular phone lines and modems for communication. Banks were early adopters of this technology on the civilian front; their central computers called each other in order to exchange transaction information.

Goodbye to punch cards

During their first decades, computers were large and expensive, and very few people were able to use them personally. Users planned their software on paper, punched them on punch cards, delivered the piles of cards to the machine room where they were run in batches, and later received the output. The means of operation changed dramatically with the introduction of terminals. Timesharing meant that several users controlled the same mainframe simultaneously from different terminals and received nearly instantaneous responses. The terminals were often connected with modems, which allowed the user to reside in a different city than the computer itself.

Technology hobbyists were also interested in computers, terminals and modems, and many of them built them by themselves. This allowed students who were technically inclined to use the mainframe from home. Before long, they also started thinking of turning their home computer into a ”mainframe” that others could connect to.

Ward Christensen and Randy Suess from Chicago were the first ones to set up a public BBS for hobbyists. The year was 1978. The system was simply known as CBBS (Computerized Bulletin Board System) and it ran on an Altair compatible. The first similar

system in Finland was set up in 1982 by Seppo Uusitupa and it was known as CBBS Helsinki.

How BBSes operated

In the 1980s, BBSes were also referred to as electronic mailboxes. During their golden age, it was difficult to explain their operation – and inherent charm – to someone who had never used one, but today, we can compare them to a website.

Imagine a site with a discussion forum. The forum has dozens of users who discuss every imaginable topic. In addition to the forum, the site has an area where you can download software and other files or upload your own files. There is also a dedicated section where you can play games against other users. You log in with a username and password, and in order to receive them, you need to answer a few questions – mostly regarding your personal information.

Most Internet users can easily imagine a site like this. Now, imagine that the site is only accessible to one person at a time. When you try to access it, it is most likely busy. It may become available in fifteen minutes or half an hour. While waiting, you can try to access another site. Once you finally connect to a site, your connection time may be limited to 30 minutes, for example. Furthermore, the connection will cost at least the price of a local phone call.

Therefore, you need to use your time efficiently. Instead of browsing through forum messages or file lists online, you should download them to your own computer. Replies to messages should be written with a specific offline reader and uploaded as a package the following night.

Of course, BBSes were not websites; users connected directly via character-based terminals. The black terminal screen slowly printed lists of commands that, when typed, allowed the user to move from one area to another and use different functions. The BBS server was most commonly located in the home of its administrator, the System Operator (SysOp). SysOps often followed what their users were doing and would sometimes open private chats with them. BBSes were often decorated with personal, colourful character graphics and, at its best, calling one felt like you were visiting the owner's home.

Teletype.

Of course, there were also larger BBSes. Some were located inside real data centres and had several phone lines – and some even charged callers a separate fee. Finland's largest BBS was MBnet, which operated between 1994 and 2002 and could accommodate over 500 simultaneous users. However, a typical BBS ran on a hobbyist's personal computer. Some were open 24 hours per day, while others were only started at night.

In Finland, calling BBSes was not very common among computer hobbyists in the 1980s. Even software pirates used mail for trading their floppies until the turn of the decade. One of the reasons was that antiquated approval legislation made modems fairly expensive to purchase and use.

BBS-style communication was also marketed for the masses: Videotex, which was born as cousin of Teletext in the late 1970s, was expected to bring data networks into every home by means of easy to use terminals. Finnish Videotex services included Telesampo and Infotel which were used for online banking in particular. However, the system was only successful in France, where the Minitel service had millions of users at one point.

```
Read Commands:

press <CR> to read next unread msg  <<> show the msg that this msg is reply to
<num> read msg with that number     <>> show first reply to this message
<+> read next message               <=> show next reply, if there is one
<-> read previous message           <0> show the first msg of this reply chain
<.> read same message again         <P> show msg that you read before this msg
<RE>ply to this message   <K>ill msg written by you  <D> add msgs to scratchpad
<PRE> private reply       <REC>over killed msg       <SE> download scratchpad
<CRE> conference reply    <MOVE> msg to other conf.  <MODE> for msg reading
<I>nfo about author       <MC>opy msg to other conf. <SH>ow conference status
<V>iew headers of msgs    <DUP>licate this message   <RES>ign conferences
<M>ark menu               <AI> info about conf.      <NEXT> move to next conf.
<S>earch menu             <SAVE> message to /tmp     <GNUS> fullscreen reader

Global Commands:

<Q> main menu             <E>nter a message          <WHO>'s on other nodes
<R>ead menu (or [ENTER])  <J>oin/resign msg areas    <CHAT> menu
<F>ile menu               <MD> download messages     </?> ext. chat menu
<U>tility menu (settings) <MU> upload replies        <H>elp about using BBBS
<B>ulletin menu           <COM>ment message to SysOp <NEWU>ser help
<G>oodbye (logout)        <TIM>e left now            cmnd -h shows command help

(Post Office:0) Read Command (? for menu):
```

Menus from a bulletin board system running BBBS.

```
                        <<< Freenetin tori >>>

  1 Freenetin toimisto/    - freenetin yll{pidolliset toiminnot
  2 Opas/                  - ohjeet, manuaalit, opashenkil!t
  3 Posti/                 - postin lukeminen, l{hett{minen jne
  4 Oppimiskeskus/         - opiskeluun liittyv{t asiat
  5 Mediateekki/           - kirjasto, tavaraa eri muodoissa
  6 Monitoimitalo/         - sekalaiset aihepiirit
  7 Uutistoimisto/         - uutiset, ajankohtaiset asiat jne
  8 Partneritorni/         - partnerien ilmoitustaulut, infopisteet
  9 Raatihuone/            - suoran vaikuttamisen kanava
 10 Kahvila/               - ajanviettopaikka, keskustelukahvila
 11 Lentokentt{/           - yhteydet maailmalle
 ------------------------------------------------------------------
  h=apua, p=edellinen valikko, m=p{{valikko, x=poistu


Olet saanut uutta postia.
Anna valintasi ==>

F1=Help | ANSI | 57600N81,1 |  |  |  | Telemate 4.21 [UNREGISTERED]
```

For many Finnish BBS users, Freenet Finland was their first contact with the Internet.

ANSI graphics from Haciend el Bananas.

One of the most popular BBS games: Legend of the Red Dragon (LORD).

Mainframes and universities

While hobbyists were setting up their first modem BBSes, mainframe operators were already busy building packet networks that used fixed connections. Since 1969, the United States had been constructing ARPANET which included military and university computers. In Europe, the French CYCLADES was a similar network, and even Finland started planning a university network already in 1974.

Fixed lines were copper lines and physically similar to telephone lines, but they were entirely reserved for connecting two points. A customer leased the entire line and received access to its entire capacity. Initially, ARPANET operated at 50 kilobits per second – twenty times faster than a standard telephone modem.

The idea behind a packet network is that the machines connected to it send out data packets that include the address of the receiving computer. The machines also continuously listen to the line and collect packages addressed to them. The intersections of the lines have routers that forward the packets according to their address. Initially, ARPANET used a packet protocol known as NCP, but it switched to Internet Protocol (IP) in 1983. At the same time, the military computers were separated into a dedicated network and the remaining part became known as the Internet.

In Finland, Internet-type communications started with networks that were entirely modem based. In the early 1980s, universities had UNIX machines that supported email messages between users (mail) and public newsgroups (news). UNIX had a program known as UNIX to UNIX Copy (UUCP) that allowed UNIX machines to transfer conversations to each other over modem connections. Mail bags were usually exchanged only once or twice per day, which meant that messages took days to travel across larger UUCP networks. The Finnish network was connected to the international UUCP network in 1983. Fidonet, which later achieved popularity in the BBS world, was also based on exchanging message packets.

Packet networks arrived in Finland in 1983 when the governmental telecommunications service started offering an X.25 network known as Datapak. The invoicing was based on the amount of transferred data. Banks were the first users of the network, but the Finnish University and Research Network (FUNET) that was started the same year also decided to start using it. In a few years, however, FUNET switched to 64-kilobit leased lines after the increase in email traffic resulted in substantial data transfer bills.

Universities had a wide range of hardware from different manufacturers, each with their own network technology, and FUNET was transferring packages from all of them. DECnet was only intended for VAX machines and other computers from Digital. EARN was a network of European universities that was established and sponsored by IBM. It offered mailing lists and instant RELAY messaging. UNIX computers only understood IP, but VMS machines could speak any protocol. The only fixed international connection ran via EARN, which meant that, even in 1987, international UNIX emails and news posts had to be transferred by modem. This situation improved the following year, when the university networks of the Nordic countries were connected to 64-kilobit lines, forming NORDUnet.

NORDUnet connected with the Internet in 1988, when a 56-kilobit satellite link was established between Stockholm and Princeton. However, the United States remained cautious about Finland, since it was right next door to the Soviet Union. The very same year, the Nordic countries lost the Internet for a week after an enterprising Finnish student had tested the security on some US Army servers.

Little by little, the Internet replaced all other protocols in the university networks. Nowadays, few people will recognise EARN, for example, even though it was the most popular data network in European universities for several years.

Worlds collide

In the early 1990s, BBSes were still very far detached from the packet networks. As a result, most BBS users had a fairly vague idea of the Internet. Some BBSes were connected to the Internet via the UUCP network, for example, and there were a couple of systems with fixed Internet connections. These were located at universities.

1993 was the year when many hobbyists connected to the Internet for the first time. Around this time, several ISPs offering reasonable prices were established in Finland. In fact, some of them were BBSes that purchased a fixed connection to the Internet and paid it back by collecting a monthly fee from their users. User interfaces were still character-based, usage times were limited and the lines were often busy.

Most services offered by the Internet had an equivalent in the BBS world. Email was similar to private messages, newsgroups were like public messages and IRC was reminiscent of the chatroom in a multi-node BBS. However, everything happened on an entirely different scale. There were thousands of busy newsgroups, some of which were dedicated to very niche topics. The chats and MUDs attracted unbelievable numbers of users from dozens of different countries, which made geographical boundaries irrelevant. Many users felt as if they had stepped from a small village into the centre of a metropolis.

Modems became commonplace in the 1990s, which could be seen as an increase in the use of BBS and the Internet. Many hobbyists used both, but for different purposes. BBSes were seen as clear, closely-knit communities were like-minded people could discuss anything and everything. In Finland, the system of local phone calls and regional phone companies created a separate BBS scene in every province. The users also arranged get-togethers with one another. The Internet was a less personal, more chaotic and limitless ”ocean of information” that was used to search for files, discussions and expertise on specialist topics.

During the latter half of the decade, the Internet started to erode the BBS world. In early 1996, there were over 500 public 24H BBSes in Finland; by the year 2000, this number was down

PTT BBS, a giant BBS in Taiwan.

to a few dozen. The availability of fixed broadband connections made many users give up their phone subscriptions and, with that, also the BBSes. The most loyal BBS users only left at the last minute, and for many of them the transition was far from easy.

Many BBS communities disbanded completely when they were unable to find a suitable gathering place on the Internet. The most resilient SysOps moved their BBSes online, either directly or by converting them to online forums, but very few of them were successful. The communities that chose an IRC channel to replace their BBS fared the best. Of course, IRC is a very different animal by nature, but it was able to capture the close feeling of community that was typical of BBSes better than forums or newsgroups, for example.

Out with the old, in with the new

The Internet has developed from a secretive elitist society into a basic human right. The researchers, students and hackers were gradually joined by representatives from all walks of life. For many, Facebook was the final step in the journey towards Internet community addiction.

The Internet has changed over the years; sometimes, this has even been for the better. Even the less technically inclined can now be heard and finding interesting content from among all the white noise is now easier. However, change always comes at a cost: when a once popular online service fades away and the users disappear, the same sense of community is never felt again. The new service will always lack a key feature and many users will never adapt to it. Communities dissolve, leaving many with an empty feeling.

Facebook will be replaced by something new within the next ten years. By then, even the general public will understand the historical nature of the Internet – how even data networks undergo paradigm shifts and how new services are always built on top of old ones. Maybe BBSes will also be given the merit they deserve?

BBS status in 2013

At time of writing, the website telnetbbsguide.com lists 350 operating BBSes. Most are only accessible via Telnet, but some American BBSes also have a phone line. There are three Finnish BBSes on the list: BCG-Box (bbbs.net), a support BBS for the BBBS software; Rampton Bird's Box (rbb.bbs.fi:32) and Haciend El Bananas (haciend.bbs.fi) which specialises in computer art.

Most modern BBSes are barren – like museum exhibits that have been left in place even though there are no more users. However, there are still some very large and active systems. PTT from Taiwan may have up to 150,000 simultaneous users.

The Western BBS scene has close ties with MUDs (Multi-User Dungeons) and the boundary between a BBS and a MUD is not always clear. MUDs are still going strong despite the WoWs and LoLs; according to mudstats.com, the largest MUDs have over 800 simultaneous players. The Finnish BatMUD has up to 200 players.

Beautiful, forgotten UNIX hardware

Many people can appreciate old home computer hardware. Older professional computers, such as UNIX workstations and servers, are not as widely known.

Story by Ville-Matias Heikkilä Images by Mikko Torvinen, Wikimedia Commons users Shieldforyoureyes, Zymos, Brian Pitts, Napoli Roma, Thomas Kaiser, Fluff, Modano, Thomas Schanz, allaboutapple.com

UNIX workstations are desktop size machines that have been designed to run a variant of UNIX. However, their background is different. While PCs are home computers that gradually took on professional tasks, UNIX workstations are scaled-down versions of mainframes.

Workstations were used for demanding professional applications that microcomputers were unsuitable for. These included industrial design, 3D graphics and scientific research, for example. Workstations were also often used as servers, and most manufacturers offered servers that were compatible with the workstation models. Later on, manufacturers started focusing solely on servers.

Cross-breeding micros and minis

In the 1970s, most computer use consisted of timesharing. However, the idea of a personal computer reared its head on two fronts: some people wanted to use cheap microchips to construct something that would barely pass for a computer, while others dreamt of machines that would dedicate all the resources of a ”full” computer to serving a single user. An early example of the latter is the *Xerox Alto* (1973), which is known as a pioneer of the graphical user interface and the Ethernet network.

UNIX was originally designed as a timesharing operating system that was used on Digital's *PDP-11* and *VAX* minicomputers via text terminals. In the early 1980s, however, companies developing single-user UNIX computers started entering the market: these included *Apollo Computer*, *Sun Microsystems* and *Silicon Graphics*. Later on, even veterans like *Digital* and *IBM* hopped on the bandwagon. UNIX had originally been developed as an open system, but it started to become commercialised and was divided into manufacturer-specific variants.

Workstations in the 1980s were commonly based on the Motorola 68000 series of processors. Its instruction set is reminiscent of the Digital minicomputers mentioned above. In the 1990s, the most common choices were 32-bit and 64-bit RISC processors that were usually developed and manufactured by the equipment manufacturers themselves. This proprietary approach also created compatibility problems between different UNIX variants.

By the turn of the millennium, standard PCs had caught up with the UNIX workstations in many respects. This caused financial difficulties for several manufacturers, and they refocused their attention on supercomputers and server hardware. Workstations went out of production or they were replaced by PC-based hardware. Decommissioned workstations started ending up in the hands of hobbyists.

So, what can you do with it?

UNIX workstations have not been viable alternatives to PC hardware for a long time. Nevertheless, they have a certain charm and air of old-age professionalism. They differ from, say, PCs running Linux in charming and bizarre ways, but are still similar enough to offer a comparable user experience. They are also small enough to fit inside a normal home.

In particular, old UNIX machines have been turned into servers on home networks. The subtleties and quirks of the hardware and operating systems make projects more interesting than standard Linux hacking. However, they can be used for other purposes as well. A good rule of thumb is that

if you can run a program on the Raspberry Pi Linux, you can also run it on a 1990s workstation.

Many of the traditional UNIX operating systems still receive version updates, since they are still used for critical servers. Finding genuine software for your machine may sometimes require a bit of luck; in its absence, you can usually test the machine with a free UNIX variant like Linux or NetBSD.

Proprietary problems

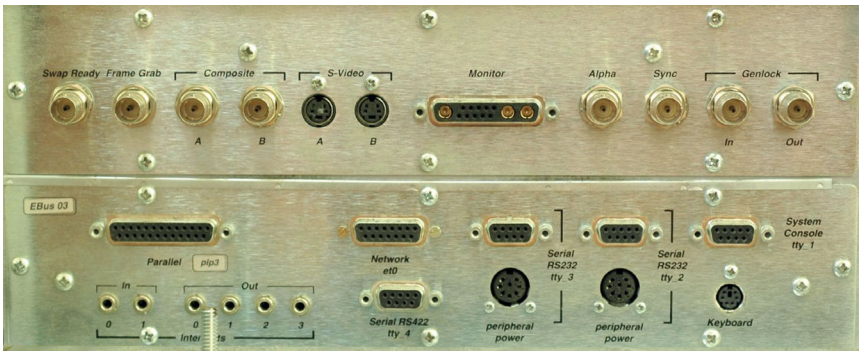
Hauling an old UNIX machine home will usually result in a host of problems. Displays commonly use a *13w3* connector or coaxial RGB – neither of which can be found on a standard PC monitor. You can either solder together an adapter or use a serial port terminal. Keyboards and mice are usually completely exotic. Hard drives, optical drives and even floppy drives are usually connected via SCSI in all models, with the possible exception of the PC-type cheap variants.

If the machine has had its hard drive wiped or removed, the first task for the hobbyist will be to find an operating system and install it. The installation may be tricky; for example, the installation CD might not boot unless the block size for the CD drive is correct. You can usually start the installation over the network via TFTP, but this commonly requires another computer with the same OS on the network.

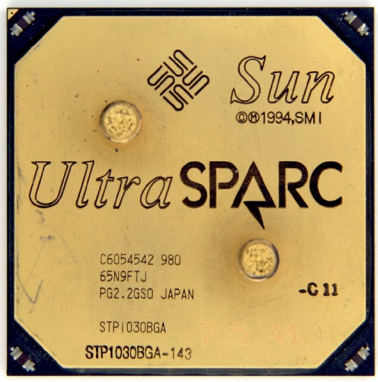
If there are any GNU software packages available, you should install them. The operation of commercial UNIX tools usually differs from that of the GNU tools, which may cause compatibility issues. The manufacturer's own C compiler is usually better at low-level optimisation for the target processor, but GCC is more compatible and superior at high-level optimisation.

You may also encounter the word *open* in commercial UNIX environments. However, this usually means that the software is independent of the manufacturer, and does not refer to open source. For example, the *CDE* desktop environment and the *Motif* library are 'open' additions to the X window manager that never became popular on the Linux side due to their commercial nature.

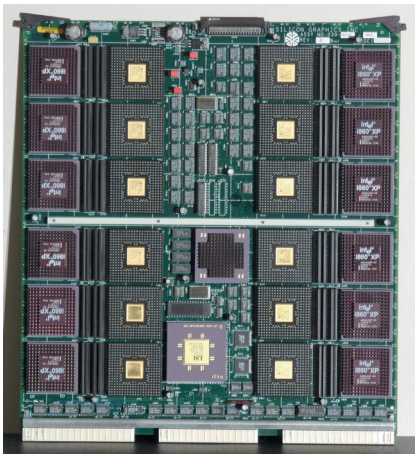
Overall, old UNIX machines are interesting devices. If you come across an old UNIX workstation, server or terminal while browsing random junk, you should definitely pick it up – they are much more interesting than old consumer PCs, anyway.



Connectors on an SGI Onyx.



Sun's 64-bit UltraSPARC 1 processor.


The Geometry Engine GE10 GPU from an SGI Onyx.


A hobbyist's collection of UNIX workstations.

Terminals

UNIX machines have always been used as servers for multiple users, and you are sure to run across terminals when hunting for UNIX hardware. Terminals can be roughly divided into two groups: There are text-based "dumb terminals" with RS232 interfaces and graphical X terminals that connect over Ethernet. Terminals usually consist of a CRT monitor, the terminal logic inside the monitor case and a set of input devices. Usually, they are easy to connect to UNIX hardware as well as modern Linux PCs.

Perhaps the most famous manufacturer of text terminals is Digital, the developer of the de facto standard *VT100*. Even though all the commonly used terminal emulators are VT100 compatible, the control codes may differ greatly between different manufacturers. However, a virtual terminal software like *screen* can correct the compatibility issues.

The X window system is built on a client-server model, which allows basically any X application to run on an X terminal, right up to the modern web browsers. Nevertheless, problems may arise due to the limited colour palettes of the old terminals and the tendency of new software to use the OpenGL API.

Sun Microsystems

Sun Microsystems was established in 1982 and the MC68000 based *Sun-1* was introduced the same year. The company switched to its own *SPARC* processor family with the *Sun-4* that was introduced in 1987.

In 1990, Finnish journalist Jyrki J. J. Kasvi described the 32-bit *SPARCstation 1* as his "dream computer" – completely unaffordable due to the FIM 100,000 (approx. $20,000) price tag. When Skrolli's Editor-in-Chief purchased the same machine ten years later, it only cost FIM 150.

Sun uses a UNIX variant known as *SunOS* or *Solaris*. Solaris and Sun hardware are held in high regard on the server side, as they can utilise a large number of processors effectively. Solaris is also available for x86 architecture and it is free for non-commercial use.

Sun has intentionally crippled its lower-end workstations in order to make the more expensive ones appear more powerful. For example, its cheap display adapters use chips that are common on PCs, but their hardware acceleration is not supported at all. The IDE disk controller drivers also run purely on the CPU.

Sun Ray thin clients were a fairly common sight on desktops in the 2000s. They could transfer sessions from one terminal to another by means of an ID card. A Sun Ray can be used for remotely accessing X window managers as well as Microsoft Windows.

The database company *Oracle* purchased Sun in 2009. The development of SPARC processors and supercomputers using them still continues.

Silicon Graphics

Silicon Graphics, like Sun, was established in 1982. The *IRIS 1000* graphics terminal was its first product, but the *IRIS 2000* and *3000* were full-blown UNIX workstations. In 1986, the company moved from the MC68000 to the *MIPS* architecture and purchased MIPS in its entirety in 1992.

Since the beginning, SGI has made a name for itself in 3D graphics, and they became especially famous in Hollywood in the 1990s. The infamous UNIX hacking scene of the film Jurassic Park shows a 64-bit *Crimson* workstation running the three-dimensional *fsn* file manager.

SGI is one of the pioneers in 3D acceleration. The display adapter on an Indigo workstation can be larger than the actual motherboard. SGI also developed the graphics hardware for the *PlayStation 1* and *Nintendo 64* game consoles, and both of them use MIPS processors like SGI's workstations. The *OpenGL* graphics API that is still considered an industry standard was originally created in 1992 on the basis of the *IRIS GL* graphics library.

The "low-cost" *O2*, introduced in 1996, is one of SGI's most popular workstations. Its higher-end models included a built-in camera. The unified memory architecture allows all of the memory, up to one gigabyte, to be used by the Graphics Processing Unit. O2s were even given out as prizes at the Assembly demo party in the 1990s.

On the server side, the IRIX operating system and, by extension, SGI hardware, has a bad reputation since it is considered vulnerable and unstable when compared to the competitor Sun.

Silicon Graphics International, technically a different company than the original SGI which went bankrupt in 2009, focuses on x86 based supercomputers and servers. MIPS and IRIX are long forgotten.

Digital Equipment Corporation

Digital Equipment Corporation manufactured computer components already in the 1950s. UNIX was originally developed on DEC's *PDP-8* and *PDP-11* minicomputers, from where it was converted to 32-bit *VAX* machines in the late 1970s.

The VAX based *VAXstationia*, which entered the market in 1984, can be considered DEC's first UNIX workstation. Although the default operating system for the VAX was *VMS*, which is completely unrelated to UNIX, DEC's own UNIX variant *Ultrix* was also offered for this machine from the beginning. In addition to VAX, Ultrix also runs on PDP-11 and MIPS based *DECstation* workstations.

In 1992, DEC introduced its own 64-bit RISC architecture, the *Alpha*. The Alpha is famous for its computing power, and even Cray selected it for its supercomputers in the 1990s. Alpha was even billed to replace the x86 architecture used in PCs at some point, which is why Alpha-based PC motherboards have found their way in the hands of hobbyists.

The UNIX variant used with the Alpha was originally known as *OSF/1*, but it was later renamed as *Digital UNIX* and further as *Tru64*. In addition to these, Alpha can run *OpenVMS* and even Windows NT to some extent.

In 1998, DEC was bought by PC manufacturer *Compaq*, which in turn was bought by *Hewlett-Packard* in 2002. The final Alpha was the 21364, released in 2004, as HP decided to concentrate on *PA-RISC* and *IA-64* architecture for its supercomputing needs.

International Business Machines

The information technology pioneer IBM, famous for its large mainframes and PCs, entered the UNIX workstation market in 1986. The *IBM 6150* or *RT PC* was based on the *PS/2* frame used on PCs, but the processor had been replaced with IBM's own *ROMP*. ROMP can be considered to be the first RISC microprocessor, since IBM developed it already in 1981. However, IBM sat on the design for years, allowing others to commercialise their RISCs.

In 1990, ROMP was replaced by *POWER*, and the workstations, servers and supercomputers using it were commonly named *RS/6000*. The little brother for the POWER was the *PowerPC*, developed in cooperation with *Apple* and *Motorola* and also used on consumer hardware.

IBM changed the branding of its system families at the turn of the millennium and the RS/6000 became the *System p*. Later on, System p was combined with *System i*, previously known as *AS/400*, which created the *Power Systems* range of servers and supercomputers. In addition to IBM's own UNIX variant, *AIX*, these machines also support Linux and *IBM i*, which originates from AS/400.

IBM is still actively developing mainframe microprocessors. Alongside the POWER processors, IBM also develops the *zEC* range which, despite its high computing power, is still machine code compatible with IBM's 1960s flagship, the *S/360*.

Hewlett-Packard

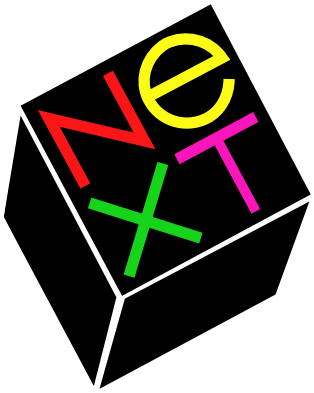
Hewlett-Packard has been manufacturing computers and calculators of different sizes and their peripherals since the 1960s. HP entered the UNIX world with the *HP 9000* range, introduced in 1984. In 1989, HP purchased *Apollo Computer* which offered a range of workstations.

In addition to MC68000s, the early HP 9000s also used stack-based *FOCUS* processors. They were phased out in the late 1980s in favour of HP's own *PA-RISC* architecture. In 2003, PA-RISC was replaced by Intel's *IA-64* or *Itanium*, and the model name that referred to the fictional HAL 9000 was also discontinued.

HP's UNIX variant *HP-UX* is still considered a swear word by many. Despite being efficient and reliable in itself, it differs from the other manufacturers' variants in several respects. HP's text terminals are not VT compatible, the default C compiler is mostly only suited for compiling HP's own code, the character sets and command arguments are different, and so forth and so on.

The most popular PA-RISC workstations were the 32-bit *712* and *715* that were launched in the mid-1990s to compete with PCs. They can accept PC keyboards, mice and displays, but are significantly faster than period PCs in terms of 2D display processing, for example.

The current machines running HP-UX are Itanium based *HP Integrity* servers. In addition to HP-UX, they can also run Linux, Windows Server and *NonStop* and *OpenVMS* which HP has acquired through business mergers. Intel continues to develop the Itanium processors.



NeXT

NeXT was established by Steve Jobs in 1985 and it had a fairly short history, but it did manage to launch a few UNIX workstations in the late 1980s and early 1990s that were notable – and not only because of their cubistic appearance.

The *NeXT Computer*, *NeXTcube* and *NeXTstation* were based on Motorola's 68030 and 68040 processors and the 56001 DSP that provided multimedia functionality. A separate RISC accelerator with a dedicated GPU was also available, but it was never properly utilised by software. Instead of floppy disks, NeXT used magneto-optical disks with a capacity of 256 megabytes.

NeXTs run *NeXTSTEP*, a UNIX operating system based on the Mach microkernel and BSD – and the other similarities with Apple's *Mac OS X* are not coincidental, either. For example, the system software is written in Objective-C, and instead of X-Windows, it uses a proprietary graphics solution with different user interface innovations. NeXTSTEP also has a variant called *OpenStep* that runs on x86, SPARC and PA-RISC.

Hidden bursts

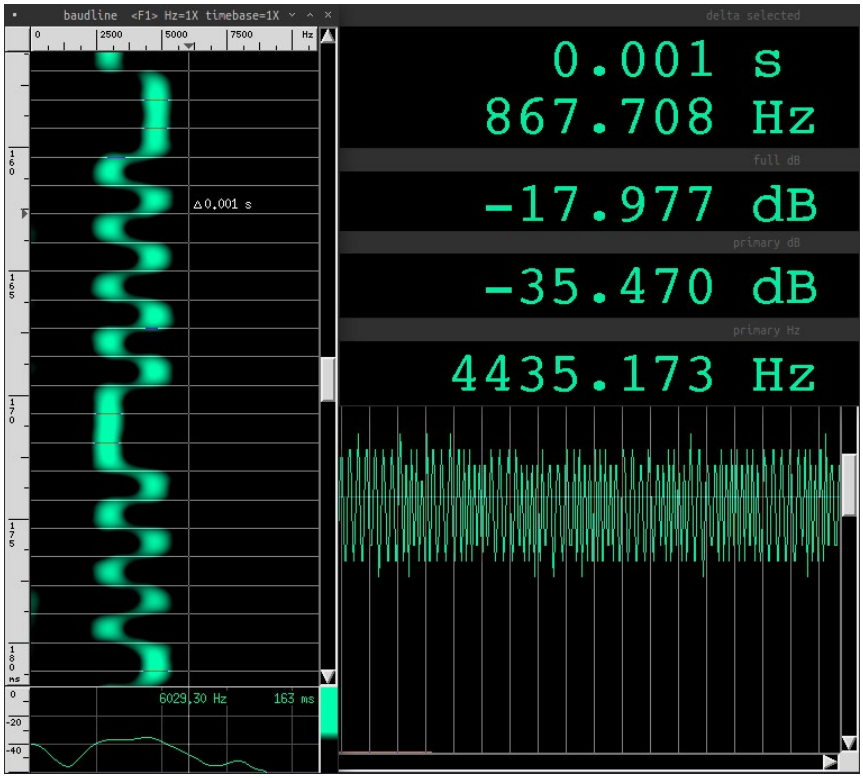
Alice has a wireless keyboard. But how safely can she use it when Eavesdropping Eve lives next door?

Story by Oona Räisänen Images by Oona Räisänen, Nasu Viljanmaa

Many computer peripherals use wireless radio connections. Since all radio traffic is fairly easy to listen to without being detected, the communications of these peripherals have usually been scrambled by means of encryption.

Out of curiosity, I dug out an old Logitech iTouch keyboard from the year 2000. It sends the keystrokes at a frequency of 27 MHz to a receiver connected to a PS/2 port. The receiver of this system is fairly large when compared to the small, modern USB receivers, and studying the circuit board will already reveal details about its operation and frequencies. The circuit board has quartz crystals at different specific frequencies and an integrated FM receiver circuit, among other things. This gives us reason to believe that the information regarding the keystroke is transmitted as modulations of the radio's carrier wave.

An easier way to analyse the operation is to listen to the radio. The 27 megahertz frequency can be heard on any portable radio that has the 11-metre shortwave band. Indeed, by placing a radio in the same room, I can hear the signal very clearly at a frequency of 27.140 MHz. It consists of bursts. One burst is heard when a key is pressed, and another when it is released. Holding the key down produces a series of shorter signals. The side of the keyboard has a button labelled "Connect". You would think that this is related to renewing the encryption keys.



Analysis of the raw sample stream.

However, the portable radio is designed for listening to radio broadcasts on the AM band and uses a low-pass filter. The filter may have removed some of the information. I can achieve a better bandwidth by using a cheap USB digital TV tuner dongle. The RTL-SDR software radio suite allows for tuning the RTL2838 receiver chip inside the dongle to an almost arbitrary frequency. I am also instructing the device to bypass its TV signal decoding circuitry and to dump the raw sample stream in real time on my computer. The sample stream is an 8+8 bit in-phase/quadrature (I/Q) representation of the radio signal. In this case, its real part can be interpreted as PCM with minor distortion and saved in WAV format, for example. This way, I can analyse and process the signal as if it were sound.

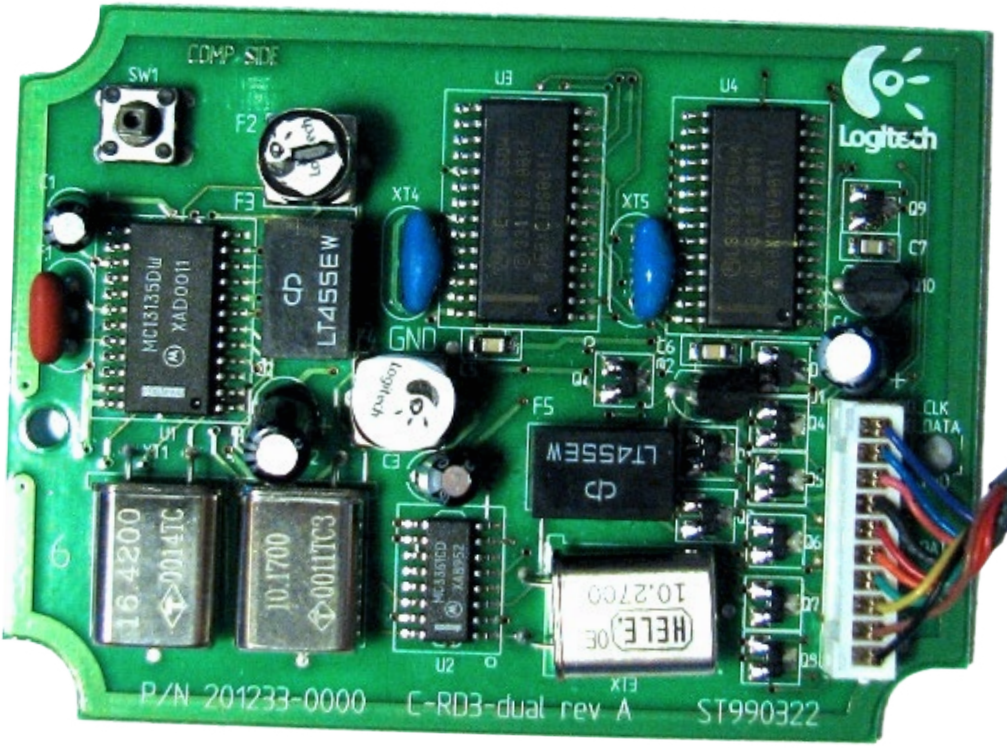
A spectrogram of the sample stream reveals that the signal is, indeed, a special application of FM broadcast: a binary frequency shift keying (FSK) with a two-kilohertz shift. In an FSK, the carrier frequency varies between two values; one of them denotes one, the other denotes zero. A rough esti-

mate based on the spectrogram would indicate that the bit rate is around 870 bps. Therefore, I use C to write a program that has an 870-hertz sine oscillator and a phase locked loop (PLL) that locks the sine wave to the data. When the oscillator reaches ascending zero, the software outputs either zero or one depending on the current frequency of the signal, which can be determined by means of Fourier transformation.

One keystroke generates approximately 85 bits of data. The length of the message varies slightly, which means that there may be some sort of encoding on top of the FSK. The first part of the bit string seems to remain the same, which suggests that it contains a synchronisation, address or header field. The end of the bit string changes between keystrokes, especially when different keys are pressed. It may contain the PS/2 key code, but the encoding is not obvious.

The communication is not encrypted in any way. This is apparent because the messages seem to match the key: pressing a specific key will always generate a bit string that is more or less similar. Even a simple encryption algorithm would hide this correlation. This also means that the Connect button is not related to a key exchange.

The unknown variable fields in the messages may be counters and checksums. They can be bypassed after a few messages have been saved for each key: this allows for creating a type example or average for the messages and saving them in a list that is linked to a specific key. After this, Eavesdropping Eve can compare the received data to all the bit strings in the list and rank the options according to their Levenshtein distance, for example. The key with the smallest distance is then printed on the screen, similarly to a keylogger.



In other words, using a Logitech iTouch keyboard for typing is comparable to shouting your passwords and email messages across the room so loudly that all the neighbours can hear them.

Of course, it is possible that the eavesdropper cannot access the keyboard in advance and try all the keys individually. However, this is a minor obstacle. For example, Eavesdropping Eve can create statistics of the frequencies of specific received bit streams and compare them to the frequencies of character combinations in a specific language, for example. This is known as frequency analysis, and it allows for determining which keystrokes correspond to which bit strings, even if the protocol is completely unknown.

Current wireless peripherals have switched from the 27 MHz band to 2.4 GHz, and encryption has, in all likelihood, also improved. Of course, frequency alone does not make listening more difficult or improve security, although the TV receiver that I am using cannot reach this band without modification. Some peripherals use encrypted Bluetooth or other common protocols. My own couch keyboard uses Logitech's Unifying technology that claims to use 128-bit AES encryption. However, its key exchange contains some suspicious stages that may warrant further investigation...

```
~/koodi/fm - zsh
2204 windy@pentti~/koodi/fm ) rtl_sdr - -f 27132000 -g 32.8 -s 96000 | sox -t .raw
-c 2 -r 96000 -e unsigned -b 8 - -t .raw -r 22050 - | ./fm |perl deco.pl
Found 1 device(s):
  0:  Realtek, RTL2838UHIDIR, SN: 00000013

Using device 0: ezcap USB 2.0 DVB-T/DAB/FM dongle
Found Rafael Micro R820T tuner
Tuned to 27132000 Hz.
Tuner gain set to 32.800000 dB.
Reading samples in async mode...
[CAPS]0wned
```

Running a program that decodes the signal.

```
nappis.txt (/home/windy)
w 1111111011110111111011110110101101110011111111100111111110111101111101111010110110000000
e 1111110101111011111101111110101011011001111111100111111101111011111101111010101101000000
1 1111111011110111111011111110110101110011111111100111111101111011111101111010101101000000
2 1111111011110111111011111110110111111100111111110011111111011110111111011110101101000000
3 1111111011110111111011111110101111010011111111100111111111011101111110111101011100000000
7 1111111011110111111011111110110101101100111111110011111111011101111110111101011101000000
8 1111111011110111111011111110110111111001111111100111111111011101111110111101011100000000
9 1111111011110111111011111110110110101001111111100111111111011101111110111101011010000000
0 1111111011110111111011111110110111010011111111100111111111011101111110111101011010000000
u 1111111011110111111011111110101011100111111111001111111110111011111101111010110000000000
i 1111111011110111111011111110101011100111111111001111111110111011111101111010101100000000
                                                                    6,8           40%
```

Bit strings from a few keystrokes.

Escaping Big Brother

While online, you always have a big brother watching you. However, he can be led astray. Learn the basics of reclaiming your privacy.

Story by Santeri Tani Images by Nasu Viljanmaa, Santeri Tani

Lack of knowledge can cause each one of us to leak our personal details online. Sharing information on the Internet is much easier and quicker than elsewhere, and this is why even personal matters can sometimes end up in the wrong hands. As a result of human error, someone else may receive a private message intended for someone you know.

The large corporations have noticed that people are unaware of their privacy and they design their services for collecting our private data. While this is a form of spying, people are – technically – consenting to it. For example, people agree to Facebook recording everything and getting to know us. This information allows advertisers to target their advertising, and selling it has become a large business.

But even the large corporations have their own Big Brothers. In 2013, the leaked PRISM programme revealed that the largest online companies have been sharing their user information with the United States Government.

Privacy and its protection have both social and technological aspects. On the one hand, it depends on what information the person is willing to share. On the other hand, different devices and software may spread information about us. Luckily, systematic spying and monitoring can still be prevented, or at least it can be made substantially more difficult. This article explains how.

Privacy starts with you

Common sense is your most important tool for protecting your privacy and maintaining control of your information. Thinking a bit about it every now and then will go a long way.

You should not disclose your own name, email or other information at every turn. Users of online services should assess which information is really needed in order to deliver the service. Of course, using your real name and email is often preferable, but they also make the user very easy to track down. In some cases, you can use a temporary email address offered by 10 Minute Mail or Guerrilla Mail, for example.

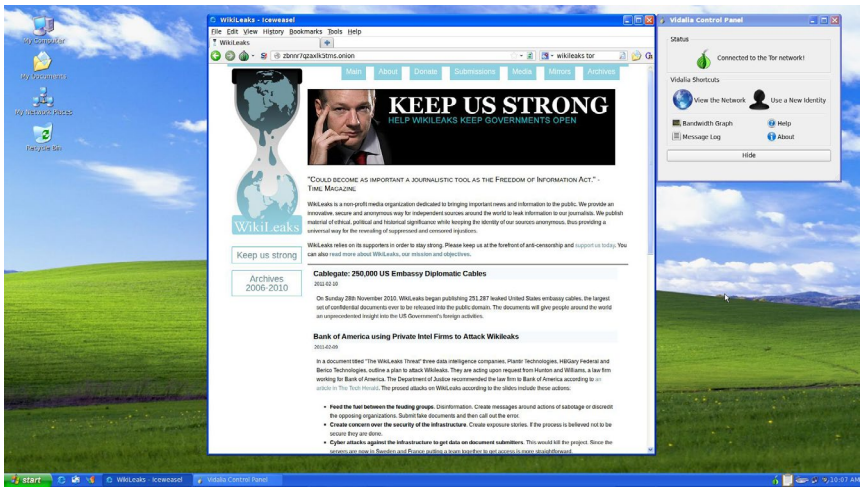
Even a regular photograph can contain information which can be traced back to the photographer. Of course, the place in the picture can be identified, but the file itself may also contain other information. Photos contain Exif (Exchangeable image file format) data that often include the model of the camera, the time the photo was taken and the GPS coordinates. These allow for determining the location where the photo was taken. Exif data can be deleted and edited with the Jhead application, for example.

The joys of free software

You should only use software that you know and trust on your computer. Before downloading software, you should always assess how reliable the authors and the source of the download are.

Software that is designed for spying on users and gathering user data is practically always closed-source, which makes studying its inner workings difficult. In many cases, modifying the software is also forbidden in the licence agreement.

Open source operating systems and software are a good starting point in



Vidalia and the Tor browser inside the Tails operating system disguised as Windows XP.

terms of privacy. Their source code is freely available, which makes it more difficult to hide suspicious code in the software.

The use of free software can be considered an ethical choice. The Free Software Foundation promotes free software and provides information regarding it. The foundation's principles are based on open source, but especially on the freedom to use and modify software. The Free Software Foundation has defined the following four principles for free software:

- The freedom to run the program as you wish, for any purpose.
- The freedom to study how the program works, and change it so it does your computing as you wish. Access to the source code is a precondition for this.
- The freedom to redistribute copies so you can help your neighbour.
- The freedom to distribute copies of your modified versions to others. By doing this you can give the whole community a chance to benefit from your changes. Access to the source code is a precondition for this.

The Free Software Foundation has created different licences for distributing software. The best-known of these is the GNU General Public License that has already become a symbol of free software. Free software is also released under many other licences, and some of them have different definitions of freedom. However, the openness of the development work and source code is the essential feature in terms of privacy and reliability. For example, GNU/Linux operating systems are generally very free, but their repositories might contain closed-source software.

For more information on the Free Software Foundation's free software licences: http://www.gnu.org/licenses/

There are also operating systems that focus on information security and privacy. These include Privatix, Tin Hat and Pentoo, all of which are good choices for users concerned about their privacy. More advanced users may want to look into Tails, Janus VM and Whonix. They are extreme privacy solutions based on virtualisation and anonymous networks.

The tinfoil hat section – how to achieve ultimate privacy

- Never let anyone else use your computer.
- Only install and use free software.
- Before installing the operating system, use a special disk such as Darik's Boot and Nuke to completely wipe your system.
- Protect your hard drive and OS by means of multiple passwords. Remember that even the BIOS password can be circumvented by removing the motherboard battery, for example.
- Use TrueCrypt, GNU Privacy Guard or an equivalent to encrypt your hard drive, home folder and all your important files. Never store the most sensitive information outside of your own head.
- Even if you are using GNU/Linux or an open-source BSD operating system, use virus protection and firewalls such as ClamAV, Snort and Chkrootkit.
- Delete files using Bleach Bit or Srm, since normal file deletion only marks the data as unused. The data itself is preserved until it is overwritten. This is unnecessary on an encrypted drive.
- Check the cryptographic signature or other certificate of any downloaded software by using GNU Privacy Guard, md5sum, sha1sum or equivalent software.
- Make sure that your browser cannot be traced by using EFF's user agent comparison tool (https://panopticlick.eff.org/).
- All network traffic should be routed through networks that protect your privacy, such as Tor, i2p, Freenet or Hyperboria.
- VPN and Tor: Only connect to Tor via a Virtual Private Network (VPN). These are available from IPredator and Tor VPN, which also accept anonymous payments via Bitcoin. You need to be able to rely on the provider of your Virtual Private Network at least as much as on the provider of your network connection. Proxies will often leak IP addresses and are unencrypted.
- Whonix: Mask your IP address by running virtual operating systems within your operating system and using one of them as a router for Internet traffic. This way, not even malware with root credentials can determine your real IP address.
- Tails: When using the Internet on other computers, use the Tails operating system that runs from DVD or USB.
- Never use wireless networks.
- Never use JavaScript, never switch on "Do Not Track" and never accept cookies. Use the IceCat browser and Jon Do Fox's privacy profile.
- If you simply must use a social network, use an alternative such as Diaspora or Friendica.
- Perform your web searches with Ixquick, Startpage or the distributed Hyperboria.
- Turn off your Internet connection when you no longer need it.

Protect your data

Using free software from a reliable source will avoid spyware with a high level of certainty. However, the computer's hard drive is used to store private data that may end up in the wrong hands by other means. What if your computer is stolen?

A variety of encryption software is available for protecting your data. The hard drive can be encrypted at the file system level, which means that all the data is saved in encrypted form. Encrypting individual files within the file system is also possible. This may require manual encryption operations.

GNU Privacy Guard (GPG) and TrueCrypt are examples of good encryption tools. The Linux kernel in itself contains encryption functions that can be used to encrypt the hard drive file system. The installers for some GNU/Linux operating systems provide the option to create an encrypted partition for storing data.

The Internet

Internet users leave behind a large number of traces. Devices connected to the Internet have an IP address that is used for identifying the device and routing data. While you are connected to the Internet, your IP address is saved in several locations, such as the databases of online stores and discussion forums. The IP address can be used to easily determine the user's Internet Service Provider (ISP) and ap-

general.useragent.override	Mozilla/5.0 (Windows NT 6.1; rv:10.0) Gecko/20100101 Firefox/10.0
general.appname.override	Netscape
general.appversion.override	5.0 (Windows)
general.oscpu.override	Windows NT 6.1
general.platform.override	Win32
general.productSub.override	20100101
general.buildID.override	0
intl.accept_languages	en-us,en;q=0.5
network.http.accept.default	text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
network.http.accept-encoding	gzip, deflate

Table 1. This makes Firefox appear as a popular standard browser.

proximate location. The police and national intelligence services can receive information on the person behind the IP address from the ISP's records (or by simply spying on them).

The PRISM programme revealed that the data traffic for all major online services is routed through the hands of the US National Security Agency (NSA). The most popular services such as Facebook, YouTube and Google may constitute the largest part of a basic user's online traffic, which exposes them to the NSA continuously.

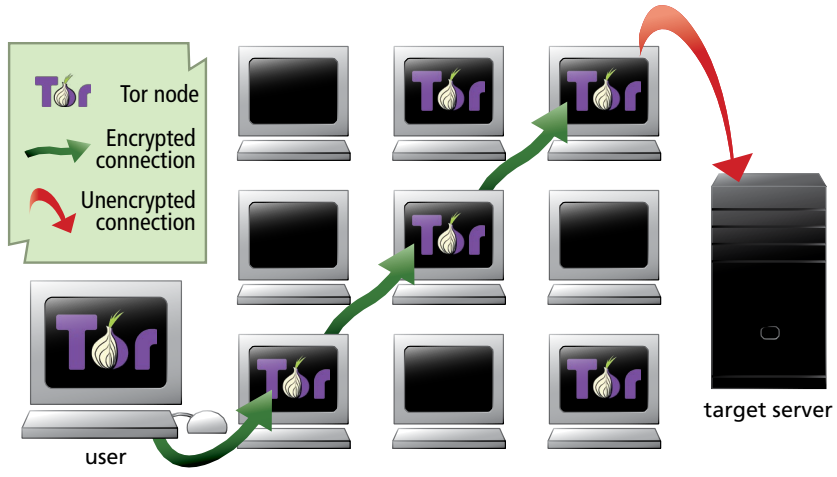
Most data traffic is unencrypted. The ISP, as well as other parties, can supervise the traffic if the communication between the online service and user is not encrypted or the connection is not made through a Virtual Private Network (VPN).

You should browse the Internet with an open-source browser such as Firefox or its sisters, IceWeasel and IceCat. Users who want a Chrome-type browser should consider the open-source Chromium.

You should disable browser history and cookies. If you need and want cookies, the Better Privacy extension is useful for managing them. Some browsers have a "Do Not Track" setting that can be used to turn off tracking for some websites. It may work, but it can also make the user's browser easier to identify.

In Firefox, you can adjust the user agent setting to make your browser and OS less identifiable. In a way, this hides the browser in the crowd and does not leave any unique clues behind.

- Typing "about:config" in the address bar will allow you to edit the settings.
- Add a new setting called "general.useragent.override".
- Enter a very generic user agent, such as "Mozilla/5.0 (Windows; U; Windows NT 5.1; en-GB; rv:1.8.1.6) Gecko/20070725 Firefox/2.0.0.6".



Structure of the Tor network.

This setting makes the browser tell the server that you are using Windows XP and Firefox 2.0.0.6. If you want to be especially careful, you can also add the values in Table 1 in order to prevent browser identification.

Websites can run scripts in your browser. Some of them monitor the actions of the user, and some can even be classified as malware. NoScript for Firefox or Scriptsafe for Chromium are add-ons that allow the user to control the running of scripts.

And, of course, you should use an encrypted connection whenever possible. HTTPS Everywhere is an add-on created by the Electronic Frontier Foundation. It checks whether the site offers the encrypted https protocol.

Instead of the official Java, consider the free IcedTea implementation. Instead of Flash, consider Gnash. Instead of using Google as a search engine, consider Ixquick, Startpage or Duck Duck Go.

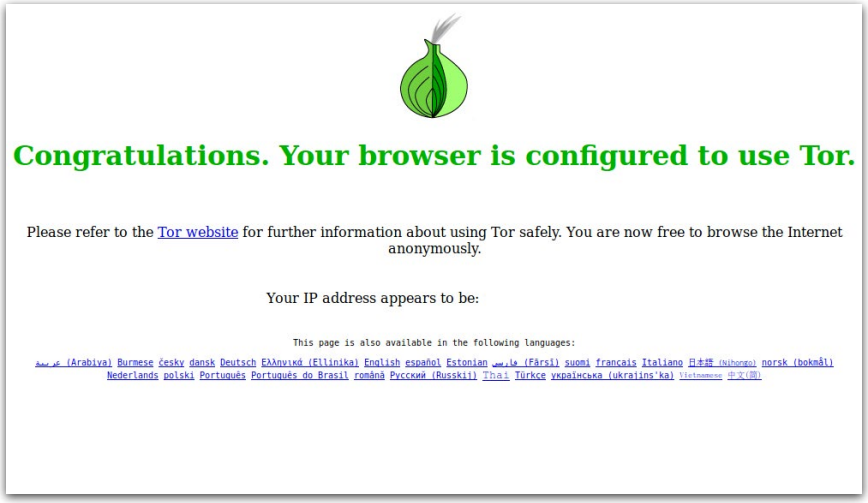
But even after all this, using a web browser will make the user fairly vulnerable. Persons who are concerned for their privacy will only use web browsers for matters where using real names is mandatory.

Tor

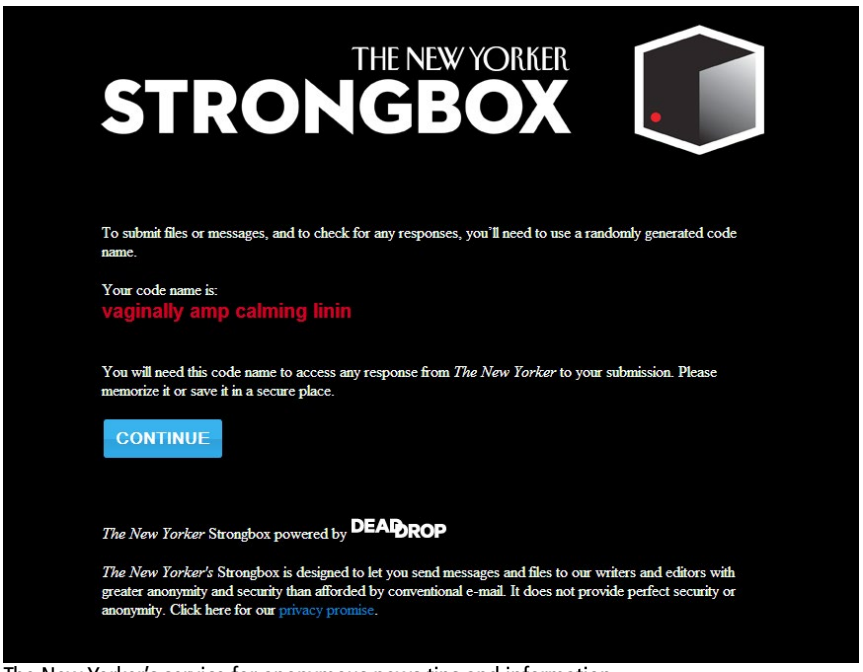
The anonymous network Tor has rapidly gained popularity in recent years. It is fairly secure when used correctly. Tor was created as a government-level communication tool and was funded by the United States Naval Research Laboratory (NRL). In 2004, Tor started to receive funding from the Electronic Frontier Foundation, an organisation promoting electronic rights, and became free software. Currently, Tor is an autonomous project that is funded by donations. The governments of the United States and Sweden are the largest contributors.

Tor nodes are mostly maintained by private individuals. There are approximately five thousand active Tor nodes, which is much too few for four million daily users. The low number of nodes and high number of users results in slow data transfer over the Tor network.

The Tor software launches a SOCKS proxy server on the computer and uses it to transfer the data. The Tor network uses onion routing encryption,



Official website for checking the operation of Tor, https://check.torproject.org/.



The New Yorker's service for anonymous news tips and information.

which is based on transferring the data through three Tor nodes. The user's computer connects to the first node and only sees its IP address. The first node only sees the address of the user and the second node, and the second node only sees the address of the first node and the final node in the chain.

Connections between nodes on the Tor network are encrypted, with the exception of the connection between the last node and the target server. None of the parties sees the entire chain. For example, the target server does not know who initiated the connection, since it seems to be coming from a Tor network node.

The easiest way to use Tor is to download the Tor Browser Bundle from the Tor Project website. It includes a Firefox browser that has been preconfigured for use on the Tor network. It also includes the Vidalia control panel for adjusting the network settings.

In addition to anonymous network use, the Tor network can also be used to set up hidden services that can only be accessed via the Tor network. In a way, Tor and its hidden services are a network inside a network. The top level domain for these services is .onion.

In order to circumvent censorship, the Pirate Bay launched its own PirateBrowser that is nearly identical to the Tor Browser Bundle but only runs through one node instead of three. This means that PirateBrowser is not a strong form of privacy protection, since the node between the user's computer and the target server can see both IP addresses.

Why Tor – are you a freedom fighter or a terrorist?

Tor has received both positive and negative publicity. Edward Snowden, the NSA whistle-blower, used Tor and the Lavabit email service to send information on the US government's PRISM programme to the press. Services such as Global Leaks and the New Yorker Strongbox strengthen Tor's position as a tool for freedom of speech, since they can be used to safely send news items to the world.

The governments of some countries regulate international Internet connections very strictly. This can make Tor an important channel for relaying sensitive information. However, identifying and blocking Tor connections is possible.

There will always be those who misuse the anonymity and freedom of speech: hidden websites have been used to distribute several terabytes of child pornography and to sell drugs. Tracking down these hidden services and their users is difficult, but the intelligence service of a country that focuses on cyber security may pull it off. The Federal Bureau of Investigation, for example, installed a JavaScript application on a server distributing illegal material in order to spy on the users of the server.

Tor and other encryption techniques may promote freedom of speech as well as unlawful activities. The technology in itself has no morals and it does not choose sides. Furthermore, it is not always clear who the villain is: the same person may be a freedom fighter for some and a criminal for others.

Useful Tor addresses

- One of the oldest sites listing hidden services, maintained by the Tor project: http://eqt5g4fuenphqinx.onion/
- A Finnish search engine containing legal .onion sites: https://ahmia.fi/
- Tor version of the Duck Duck Go search engine: https://3g2upl4pq6kufc4m.onion/
- Privacy-driven email service URSS Mail: http://f3ljvgyyujmnfhvi.onion/

Why demos suck

You cannot talk about Finnish computer culture without mentioning the demoscene and its productions. However, the essence of demos is not often understood. Skrolli will now attempt to explain how you should approach demos in order to get something out of them.

Story and images by Ville-Matias Heikkilä (Viznut/PWP)

Finland has the most demosceners per capita in the whole world. Most of Finland's best IT experts have a background in demos, the Finnish game industry is more or less dominated by demosceners, and the most significant computer hobbyist events were originally demoscene gatherings. Very few can appreciate demos as an art form, however. How could blobs that bounce to the beat of a monotonous tune be interesting? Why is no one interested in a deeper message and plotline? Are demos actually worthwhile to anyone who is not a member of a secret society?

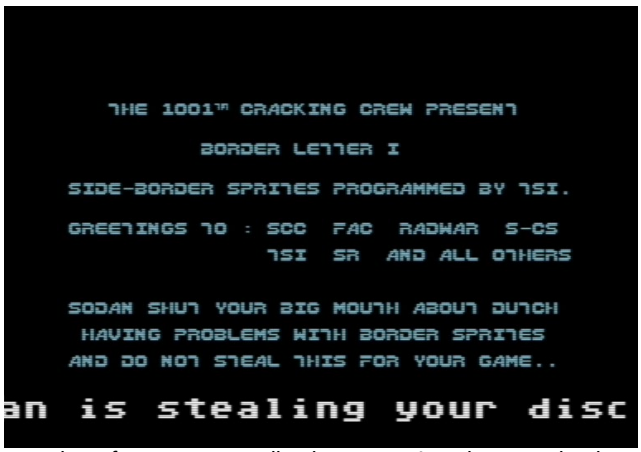
Talent contest

Demos were born when young computer hobbyists wanted to showcase their skills. Games already had copy protection in the early 1980s, which made the ability to crack them a valuable asset within the community. And, naturally, you had to sign your own work. The crackers started by editing their initials into the texts in the games, but before long, creating stylish signatures also became a dedicated sport. This gave rise to crack intros and separate demonstrations of skill – demos.

In earlier demos, in particular, most elements can be purely classified as bragging: "Look at me, I made this!" Those who set new records could also say "Look at me, this is possible!" The viewing experience is affected by your understanding of the technological framework involved, as well as the knowledge of what has been done earlier within the same limitations. The uninitiated might be in awe of an average four-kilobyte demo, since they have no prior experience of what has been achieved in this file size.

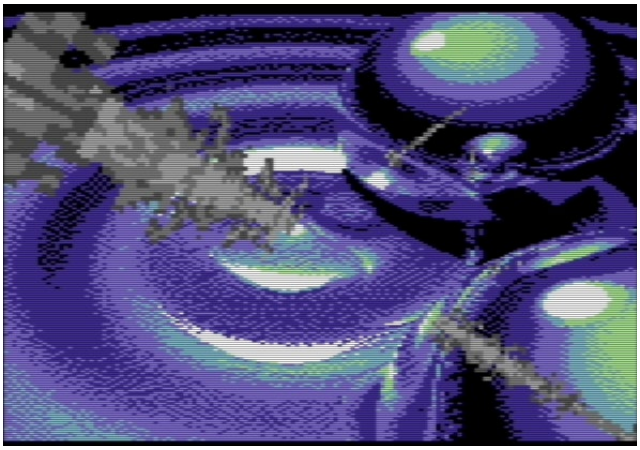
Pitting one's skill and achievements against others still remains an important motivator for making demos. Even though the age of testosterone-filled gang wars is behind us, most demos are still released at demo competitions, or 'compos' for short, where the winners are selected by public vote. Of course, there are many ways to appeal to a crowd, but breaking technological barriers with style continues to work well.

However, the limits of modern computers are difficult to reach, and the game industry with its million-dollar budgets also raises the bar for its part. Over the years, the technical innovation within the demoscene has shifted from the multiple-megabyte demos towards stricter categories that typically limit the size of the executable, the platform, or both. An executable of a few kilobytes in size cannot store an enormous 3D landscape by traditional

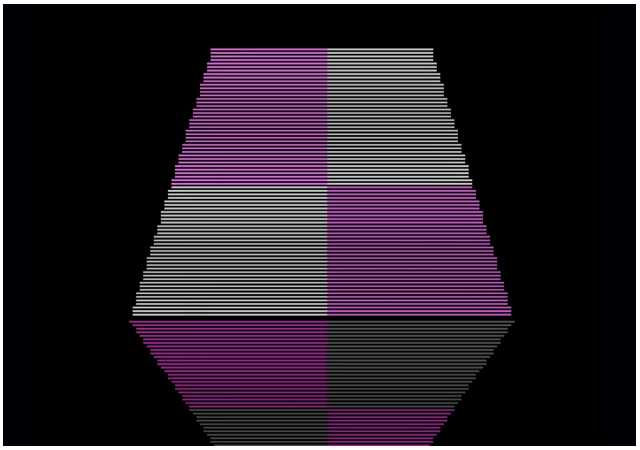


C-64 demo from 1986: A scroller that crosses into the screen border? See it to believe it! (1001 Crew: Border Letter I)

C-64 demo from 2008: A plasma effect that crosses into the screen border? Yawn, another raster timing exercise! (Booze Design: Edge of Disgrace)



What better way to ridicule the PC scene than to recreate its most overestimated demo on the C64. (Smash Designs: Second Reality 64, 1997)



Once all the standard platforms have been conquered, it is time to up the ante. (Trilobit: Doctor, Atari 2600, 2008)

means; it needs to be generated procedurally. On 8-bit machines, on the other hand, a universal 3D engine will not get you far. In order to accomplish new things, you need to utilise the characteristics of the hardware in unconventional and creative ways. Someone once compared demosceners to ninjas: they simply must use the window even when the door is in plain sight.

The importance of limitations and challenges has created a unique relationship between the demoscene and different computing platforms. In traditional computer culture, hardware characteristics always finally boil down to computing power, and old computers are replaced by newer, more powerful ones. For demo writers, however, all platforms exist here and now, each with their own special features and limitations that give rise to specific challenges and aesthetics. In addition, the platform choice is influenced more by personal taste than the age of the author: hobbyists born in the 1990s, for example, may prefer the Commodore 64 for their demo art instead of a modern PC, and nobody will consider this odd in any way.

Value in beauty

Technical achievements are short-lived, however. Someone will always advance the technique, making old achievements less spectacular. Demos that are enjoyable year after year and decade after decade are also aesthetically pleasing: they offer good music, beautiful graphics and smooth transitions. Most modern demos, in particular PC demos that have very loose limitations, abandon all technical flamboyancy and are built solely on aesthetics. In other words, you can enjoy demos without any in-depth knowledge of their inner workings.

Demos have had the aesthetic dimension since the beginning. Many long scroll texts from the 1980s would have been left unread if they had not been accompanied by good music and beautiful raster bars. You could keep the same part running for hours, since the user could move to the next part by pressing the spacebar, for example. The 'trackmo' style became dominant in the early 1990s; it combined music, visual displays and loading new content from disk into a music video-like experience that lasted a few minutes. Most modern demos still adhere to the trackmo structure.

Typical demo aesthetics are usually not very easy to approach. Some people try to view them as short films, which makes them appear scattered and lacking in content. Those who approach demos like music videos or VJ sets can usually get more out of their visual style. In any case, watching demos takes some getting used to. It is like learning a new genre of music that has visual instruments in addition to the audible ones.

A genre is also a good point of reference since most demos are very focused on form. The same standard elements and stylistic touches are repeated from one demo to another: sending greetings to other groups, rotating cubes in honour of old traditions and syncing everything to the beat of the bass drum. Of course, many groups have their distinct style, but the genre features are nevertheless hard to miss.

In part, this focus on form is due to the competitions where the works are ranked. An experimental demo might

Landscapes are a mainstay in miniature-sized demos. Nowadays, the entire landscape is generated in shader language. (RGBA, Elevated, PC-4K, 2009)

Many have considered this Russian demo humorous, but the author is serious about their political message. (Cyberpunks Unity: R, ZX Spectrum, 2004).

When is this going to start? I don't get it; it should be disqualified! (Halcyon: Chimera, PC, 2002)

not succeed, and overly emphasising content may be seen as a questionable attempt to beat another entry that demonstrates higher technical skill. On the other hand, many of the scene veterans also suffer from a lack of ambition – for them, simply doing something in order to prove that they are still active is sufficient.

Do we even need substance?

In addition to technique and aesthetics, many demos also seem to include actual substance: themes, plotlines and even messages. Mostly, however, this is all smoke and mirrors. If the audience consists mostly of people unfamiliar with demos, it is a good idea to take the aesthetics in a more cinematic direction. This makes it seem as if the demo is telling a story. However, the depth of this narrative is usually comparable to Italo-disco lyrics.

Whether demos are an art form was already under discussion in the 1990s. For many, demos are more comparable to handicrafts; they are exercises in technique that create aesthetics within the limits set by technology. On the other hand, there are also demos that emphasise artistic presentation and only use technology as a medium. Most demos are somewhere between these extremes, but usually closer to handicrafts than art.

In principle, the technology behind demos offers an excellent platform for content-driven works. However, turning your idea into a demo does not usually make sense unless the choice of technique in itself is a part of the message. Someone wishing to make animated films, for example, will be better off learning an animation suite than coding demos. On the other hand, if your idea is so unusual that no existing technique seems to suit it, demos may be a viable platform.

The limits of expression

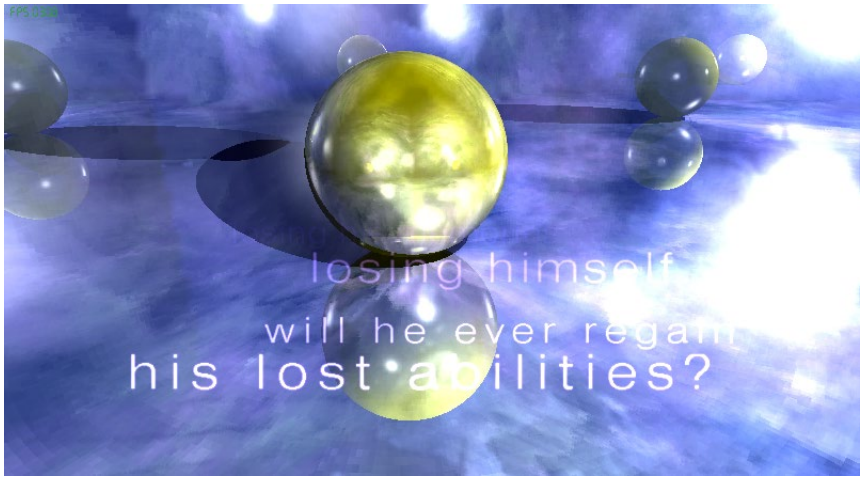
The common conception is that breaking technological boundaries is the unwavering foundation for all demo art. Technology is always taken to new extremes, achievements are compared and the innovators of new tricks rise above all the others. Content, form and aesthetics are most often considered by-products and their boundaries are not broken as eagerly.

However, pushing the limits of expression has always been a part of demo art. The current demo aesthetics would have never been born if the Amiga groups in the 1990s had not broken preconceptions concerning what demos can be. If a demo differs too much from the stabilised norms, it may be very poorly received. In order to protect their reputation, many demo authors publish their more avant-garde demos under a different alias than the one they use for the "serious" productions. Demos with extreme styles often draw upon 'noise' or 'glitch' aesthetics and use an ambient soundtrack in place of more conventional styles. Only the imagination is the limit for these experiments.

However, there is one standard that even the most avant-garde demo authors will not touch – the fact that demos are exactly the same on each run.

The main reason for their stubbornly static nature is probably that they are designed for single showings during demo competitions. Nothing must go wrong during the execution, which makes minimising the uncertainties a sensible choice. Code that operates at the technological extremes may be prone to errors. Creating a fixed run scenario can help to remove some of them. Furthermore, most demo programmers also write code for a living, which makes it understandable that they do not want to focus on bug-hunting and handling exceptions.

Demos would probably be more random and interactive if their main venue was still the home computers of individual hobbyists. Even a small amount of dynamism could also be used to explore the large grey areas that fall between demos, games and software toys. This might uncover entirely new barriers to break. However, the main motivation for demo authors are the demo events and their standardised competitions, and few people



A beautiful, technically impressive 64K PC demo with a ray-tracer routine – but did you really need to ruin it with bad poetry? Luckily, it can be switched off. (Exceed: Heaven Seven, 2000)

feel the need to publish anything outside of these events.

Some people will never get it

Demos may suck for a number of reasons. Their aesthetics are hard to grasp, their formulaic nature is dull, there is no real content behind the facade and the inside jokes fly over your head. The technical achievements are hard to understand or appear petty. Although most demos are poor, even in the opinion of their authors, the uninitiated find it difficult to appreciate even the good ones.

The cliquey atmosphere, which is also typical of other subcultures, has been a blessing and a curse for the demoscene. On the one hand, it has allowed demo art to develop on its own terms and in its own direction and, on the other hand, it has also made demos difficult to approach and very formulaic. At their worst, the scene's internal standards prevent demos from finding new paths and their authors from seeing further.

The problem with cliques has been acknowledged for a long time. At the turn of the millennium, when the demoscene had already ceased its rapid expansion, demo art entered the mainstream. The scene started connecting with the art and science circles, writing books and arranging demo shows and seminars. The Alternative Party, an event arranged since 1998, has questioned the standards of the demoscene and tried to find fresh views on demo art.

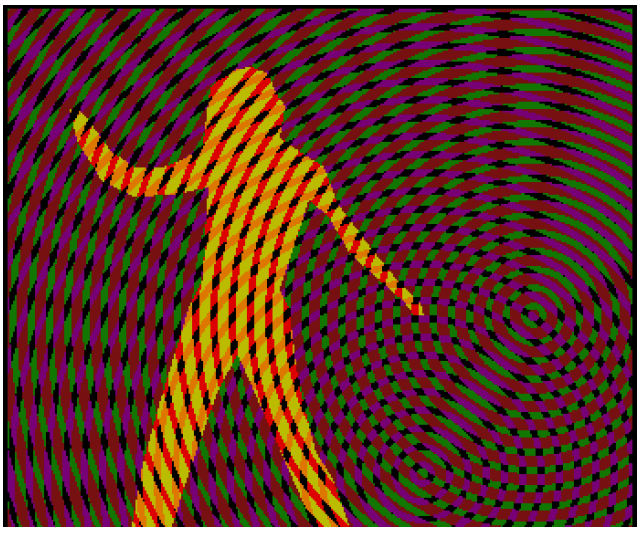
Although the efforts have been fruitful, the basic problem has remained: making demos is largely an inside hobby that is difficult to place inside an external framework. Demos are too shallow for art, too eccentric for entertainment and too technical to attract academic interest. Gamers cannot play them and hackers cannot read their source code. Demo authors may find it difficult to discuss the topic with people who do similar things for different reasons. In order for demo art to receive the approval and appreciation it deserves, we need research that defines its relationship with the rest of the world.

Academic research into demos has matured in recent years as researchers have found theoretical concepts that are suitable for describing them. For example, Daniel Botz, who wrote his dissertation on demo aesthetics, has found that, for demo authors, the different platforms are not so much tools as raw materials for forming their works. Digital media researchers Nick Montfort and Ian Bogost have set up a new field of research, platform studies, which may prove highly useful in demo research, as well.

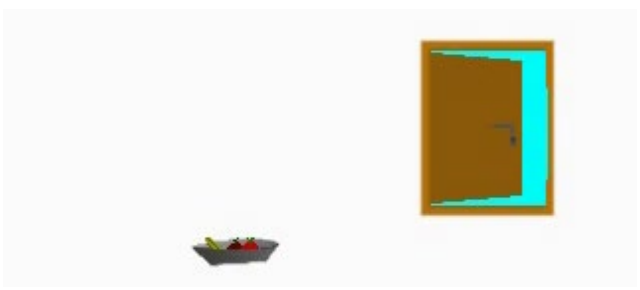
Demos are a unique form of expression that offers limitless possibilities and is guided by radical technical experimentation. However, traditions and formalities are limiting their potential. It would be beneficial for the future of demos if people made, watched and understood them even outside of the traditional demoscene. This is why Skrolli will be discussing demo culture in the future, as well.

Greetings must always be sent in traditional fashion, even if your demo is built around a hardcore punk soundtrack. (Traktor: Jesus Christ Motocross, Amiga, 2009)

At first glance, this seems to be saying something, but in reality, it is a fairly incoherent series of miscellaneous cool stuff. (ASD: Lifeforce, PC, 2007)



In its time, even this classic was frowned upon, as it favours content over technical achievement. (Spaceballs: State of the Art, Amiga, 1992)



Not even the holy cow of these skill competitions is safe from criticism. (ISO: Vati, PC, 1997)

Fail! HOW SEGA DROPPED THE BALL

In 1992, Sega held 65% of the U.S. console market. Less than ten years later, it withdrew from the hardware market entirely. How was this even possible?

Story by Mikko Heinonen Images by Wikimedia Commons/Evan-Amos

Sega originated from a company called Service Games that was established in Hawaii in 1940. After World War II, the founders, Raymond Lemaire and Richard Stewart, moved operations to Japan, where the United States had set up several military bases. Service Games imported jukeboxes and other entertainment from the States for the enjoyment of the military personnel stationed there. The company merged with a competitor, Rosen Enterprises, shortened its name to Sega and soon found that it had become a manufacturer of entertainment devices instead of only an importer.

The arcade business was booming in the 1970s, and Sega grew alongside it. However, arcades lost momentum in the early 1980s, and falling revenues scared the company's American owners. The Japanese branch started to gain more influence. At the same time, the company turned its attention towards the consumer market, and its first console SG-1000 (Sega Game 1000) was launched to compete with the Nintendo Famicom in 1983.

The SG-1000 was not a success, but Sega continued developing it. Through various iterations, it finally became the console known as the Sega Master System. Even though SMS was unable to dethrone Nintendo, it did find a following in Europe and South America and lived until the 1990s.

MegaDriving it home

The 16-bit Mega Drive (Genesis in the US) was Sega's only undisputed success. Its design drew on the company's finest expertise. Sega's System 16 arcade hardware had already demonstrated the capabilities of the combination of Motorola MC68000 and Zilog Z80, and by 1989, its price had fallen to mass-market friendly levels.

The company wanted to ensure success in the United States, in particular. Sega had ditched its old marketing partner, the toy company Tonka, and built Sega of America from scratch by recruiting the best people in the business. It had also offered the Genesis to Atari, but the Tramiels were not keen on the idea. This was probably for the best. Across the pond, the company purchased the publisher Virgin Mastertronic and turned it into Sega Europe.

Mega Drive was launched two years before its main competitor, the Super Nintendo. Its games looked especially good alongside the ageing NES, and Sega of America's marketing milked this to the last drop. *Genesis does what Nintendon't* was a catchy phrase, but sales were initially unconvincing: only half a million consoles were sold during the first year.

Halfway through 1990, Tom Kalinske was appointed the CEO of Sega of America, and he immediately set things in motion by hiring top people in his management team, lowering the price of the console and launching the development of games that were especially designed for the Western market. These were smart moves that brought Sega a 65% share of the 16-bit market in North America. Sales were also healthy in Europe, and the console sold approximately 29 million units during its lifetime. This made the lacklustre success in Japan easier to tolerate.

Now what?

In order to maximise its profits from the Mega Drive, Sega started to innovate different accessories. CD-ROM was a buzzword, so a CD add-on sounded like a good idea. The Mega-CD (Sega CD in the States) saw the light of day in late 1991. In addition to a CD drive, it also added graphics processing power, but failed to attract buyers. One of the reasons was that most games were grainy full-motion video affairs with very little actual gameplay. Fewer than 10% of Mega Drive owners

bought the CD add-on.

The Mega Drive was also licensed to other manufacturers. At least Aiwa, JVC and Pioneer made their own versions with special additions. Aiwa's version doubled as a radio and CD player, the JVC had a karaoke function and the Pioneer LaserActive played LaserDisc movies. It even had a few LaserDisc games. Sega also produced several variants, such as the Mega Jet designed for use on board airliners, but none of them became very popular.

Stormy weather on Saturn

Most successful companies struggle with finding the next big thing, but very few go as far as Sega did. The opposing parties were Sega of Japan, the cornerstone of the entire company, and Sega of America, the creator of its greatest success. The topic was, of course, how to master the next generation.

Sega's R&D was traditionally based in Japan. There, the company's engineers had developed the Virtua series of coin-operated games that were groundbreaking in the early 1990s. However, their technology was far too expensive for home use, so Sega had launched a project with Hitachi for the development of a new RISC processor. The new console would be known as the Saturn, and it would include two parallel SH-2 processors and a host of other chips.

The Americans considered the architecture unfeasible. Kalinske contacted Silicon Graphics and asked for help in designing the graphics chip for the Saturn, but Japan called off the deal. Soon afterwards, SGI teamed up with Nintendo to create the Nintendo 64, a substantially more successful device. It was also rumoured that when Sony and Nintendo fell out over the design of the Super Nintendo CD-ROM expansion known as the Sony PlayStation, Sega of America had tried to negotiate a deal on the hardware. However, Tokyo was adamant that no outside help would be enlisted. Nevertheless, after seeing the PlayStation prototype, Sega's management decided to revise the design of the Saturn and add more 3D processing power.

32 times the misfortune

The development of the Saturn was proceeding slowly and competitors were launching new products. As a stopgap solution, both halves of Sega decided to turbo-charge the Mega Drive. An add-on known as the 32X was created; it included RAM, two SH-2 processors and a selection of other hardware. The unit, which was placed on top of the 16-bit workhorse, required a dedicated power supply, but it could utilise the controllers and possible CD drive of its host. Cartridges would be the preferred format for delivering games, however. The goal was to launch the product on the US market for the 1994 holiday season in order to compete with the 3DO and Jaguar, among others.

Sega of America was left in charge of the 32X, while Sega of Japan focused on finalising the Saturn. The work suffered from chip shortages and the fact that documentation for the processors was only available in Japanese. The real bomb dropped, however, just as the device was being launched. Sega of Japan was bringing the Saturn to the Japanese market on 22 November 1994, one day after the US launch of the 32X. Even though the 32X shared some chips with the Saturn, it had no software compatibility, and the Saturn had no slots for the 32X's game cartridges.

Sega of America had been given a mountain they could not climb. First of all, the entire device was now pointless in terms of overall strategy. Secondly, the consumers were wondering why it was even being offered to them when an entire new console was only a few months away. Against this background, sales of fewer than 700,000 sold consoles and 40 published games could be considered a win, but in real life, this was a bitter and expensive disappointment. The add-ons were soon in the bargain bins and American consumers, in particular, had very nearly had enough of Sega's useless product introductions.

In the shadows of the PlayStation

The US launch date of the Saturn was set at 2 September 1995, one week before the PlayStation. However, Sega of Japan panicked and wanted to bring the launch forward, despite the fact that the Genesis was still selling. A plan was hatched where Sega contacted a few distributors and agreed on advance sales of 30,000 units in connection with the E3 expo in May 1995.

Sega's surprise attack was a complete and utter disaster. Other distributors were not informed, which caused many of them to become very upset. KB Toys, a large retail chain, terminated its contract entirely. Buyers of the advance units were only offered a few games made in-house at Sega, since not

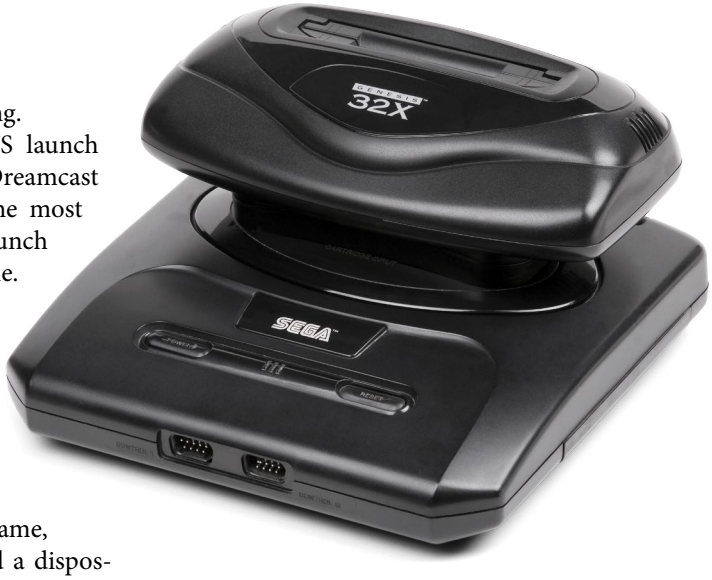
even third-party developers were told about the changed plans. This meant that there were no queues to speak of. The final nail in the coffin was Sony's announcement at E3: the PlayStation would launch at $100 below the price of the Saturn. Sony's cheaper console would be the more popular choice. At the same time, launching the Saturn efficiently killed off the sales of the Genesis, which had already been waning.

After this battle, the war had been lost. The Sony PlayStation decimated the Saturn on the US and European markets. Japan was the only place where Sega could compete at all. Total sales of less than 10 million meant that the Saturn was the weakest of the big three: Nintendo 64 shifted 30 million units and PlayStation broke 100 million. The complex architecture of the Saturn made cross-platform ports difficult and popular 3D titles looked worse on the Saturn almost without exception. Sega's own quality titles only went so far.

Sega lost hundreds of millions of dollars in the Saturn project, and its management was also in turmoil. Tom Kalinske left the company completely, and even the executives at Sega of Japan switched tasks in rapid succession. It was evident that the company could only make one last effort.

An innocent casualty

The Dreamcast was the successor to the Saturn and quite probably the best console ever developed by Sega. It might even be one of the best consoles ever, period. Its development was not free from drama, however. Sega had originally partnered with 3dfx for the development of the graphics chip, but the plans were leaked. The PowerVR2, which was chosen as the replacement, was still very adequate for 1998, however. Furthermore, Sega also brought a solid line-up of its own games to the Dreamcast. This makes it all the more tragic that the Dreamcast also carried the burden of Sega's old mistakes from the very beginning.



Nevertheless, the US launch was excellent, and Dreamcast held the record for the most consoles sold on launch date for a long time. Pre-orders alone accounted for over 300,000 consoles. There was even some positive buzz in Europe, but Japan remained unimpressed. This was a shame, since the Japanese had a disposable income that Sega could have used.

Sales were not as great after the launch, since the earlier mishaps had alienated some of the resellers, as well as the publishing giant Electronic Arts. The lack of EA's annual sports titles caused many buyers to abandon the machine, even though Sega did make an effort to replace them with its own Sega Sports range. Buyers and resellers were still bitter about how quickly Sega's previous products had become obsolete.

The Dreamcast was finally killed by Sony's advance marketing of the PlayStation 2. A few years ago, the Saturn had been destroyed by the original PlayStation, and now, the mere promise of a new version was enough to stop Sega's marketing in its tracks. The company had depleted its cash reserves with the Saturn, and it had no salvos left to counter Sony's media blasts with. Moreover, the Dreamcast was missing a DVD drive, which many buyers wanted in their next-gen console. After posting huge losses for three consecutive years and selling ten million Dreamcasts, Sega threw in the towel in January 2001. The DC would be killed off and the company would resume operations as a game publisher.

An honourable end

The old Sega went down guns blazing. Going all in on the Dreamcast ensured that it will be fondly remembered by gaming historians. The company used up its last resources to write a swan song that had a lot going for it. The original game series created for the Dreamcast saw sequels on other consoles, and Sega finally turned a profit by selling them on Sony, Nintendo and Microsoft hardware.

However, a feeling of wasted potential is also inevitable: the Dreamcast showed what Sega could do when its different departments were not fighting each other. Many players consider it to be the last hardcore games console. The Sega Saturn, on the other hand, is probably the only system that failed so hard it also killed off its technologically superior successor.

chilicorn.org

Salvaging viable hardware

Electronic devices are commonly discarded when they become obsolete. However, a little recycling and repairing could extend their service life.

Story by Albert Laine

Images by Nasu Viljanmaa, Andrew Gryf Paterson

Our everyday life is assisted by a number of devices. One of them wakes us up, another prepares our morning coffee and we use a third one to travel to work. Devices allow us to receive and share information, inspiration and ideas.

Humans have a unique ability to build tools and boost productivity. This characteristic has created an environment that we ourselves have shaped – civilization. Our way of life also tends to create more and more objects. Obsolete and faulty objects are discarded even though able hands could give them a new life.

The healing power of a hot air gun

I have spent a decade using hardware that others have discarded. The workstation that I am writing this article on has been salvaged from the dumpster. Teenagers had been using the computer as a target for paintball practice. It was missing a hard drive, but all the other components were still inside. The computer is a Dell Optiplex 760 with a quad-core Core 2 processor, a GeForce GTX 460 graphics card and 8 GB of DDR2 RAM. It is by no means the latest in hardware, but you cannot beat it for the price. And when I found it, it was only two years old. Not bad!

When I first tested the computer, it booted up, but turned itself off after a few minutes. I also tried another power supply and got the same result. After spending some time thinking about the problem, I came to the conclusion that one of the memory modules must be damaged.

Running Memtest revealed that one of the 2-gigabyte DDR2 memory modules went haywire after it warmed up. I went online to find a solution to the problem. I decided to try a popular method for homebrew repairs of graphics cards and game consoles – baking them in the oven. However, instead of placing the entire computer in the oven, you should use a temperature sensor and some tin foil to insulate the electrolytic capacitors and plastic parts that cannot withstand a temperature of 200°C (390°F).

I decided to use a hot air gun to heat the broken memory module. I placed the memory module on two blocks of wood and started heating it at a low temperature. I gradually increased the temperature to the target of 200°C. The memory module suddenly let out an audible snap, and I was quite sure it had died. Nevertheless, I decided to test it on another motherboard. I let Memtest run for a few hours and, to my delight, I noticed that it was operating properly. I transferred the module back to the machine I found – and have been using it to this day!

This was clearly a manufacturing defect, as I have not come across a memory module with a similar problem since then. The Internet had pointed

Cindy Kohtala and some loot from the Sortti recycling station.

me in the right direction, but there were no guides available for reviving RAM modules with hot air.

Successfully repairing the memory module made me try the same method with other components that I had found. The technique works with graphics cards that display checkerboard artefacts or have no display output. I have also used it to revive numerous laptop motherboards. Many GeForce GTX/GTS 8800 graphics cards, which offer high computing power, have the same issue.

I have put together a total of 13 dual core computers from components that I have salvaged and repaired, and equipped them with graphics cards offering 500–1,000 GFLOPS of performance. The server farm in my office has never let me down; every job has been completed before the deadline.

Game consoles, laptop computers and graphics cards commonly use Ball Grid Array (BGA) surface mount components. My experience is that these are often damaged due to the European Union's RoHS directive. The regulations forced manufacturers to switch to lead-free solder around 2006. Unfortunately, however, the high temperature variations present in powerful components will break the solder joints after 2–3 years – just in time for new technology to enter the market.

In the end, the regulation that was intended to preserve Mother Earth ended up hurting it by multiplying the amount of waste electronic equipment and increasing the manufacture and sales of new devices. The manufacturers won again.

The power of duopolies

A duopoly is a common situation in a capitalist market economy. It refers to a situation where two competing companies offer the same service or product. In practice, two market leaders communicate with each other in order to divide the market and to synchronise the releases of new products. Despite major differences in architecture, the manufacturers offer similar performance in the same price range.

Technological innovations and manufacturing methods advance in leaps. The design of a new type of Graphics Processing Unit will take years, for example. The company also needs to generate an income during the development, and steady sales are a requirement for this. A few years ago, the GPU manufacturers nVidia and AMD were sued due to the suspicion of duopoly-like cartel agreements. There were strong suspicions of market manipulation, and the accused settled the case out of court with millions of dollars. Since there was no court decision on the case, the activities were allowed to continue.

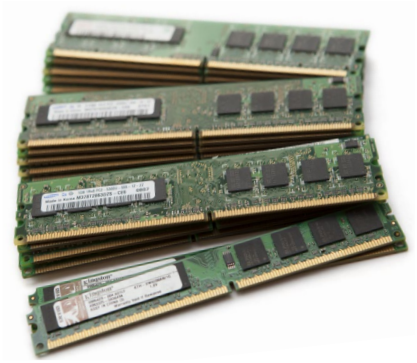
Abandoned but powerful graphics cards. Pictured: GF9500, 8800 GT, Sapphire HD 3650, 8800 GTS and 9800 GT.

Consumers have poor memory. Yesterday's state-of-the-art technology will soon find its way to the already enormous scrap heap. A great example of this is the hardware decoding of H.264 video. Graphics cards have had this feature since 2004. The GeForce 6600, for example, has hardware H.264 decoding, but it only works with a specific driver version and a specific version of specific video playback software. Once the customer upgrades the drivers, playback software or operating system, the feature stops working. This made the computer sluggish when viewing high-resolution online videos, and caused many consumers to purchase a new device. If the decoding support had been retained, there would have been fewer buyers for new hardware versions.

Fix-mod-hack!

I started my life with consumer electronics in the same way as everyone else: as a consumer. When something broke, it was taken to the shop for service. I became interested in the oper-

A server farm made of salvaged hardware. Yesterday's powerful PCs work well as a team. The workstation has a RAID disk server connected over eSATA.



You can easily scavenge piles of usable DDR2 memory modules from discarded computers.

A selection of expansion cards. In the foreground, a laptop graphics card (HD3650) and a desktop adapter for the PCI-E-bus found on laptops; Wi-Fi adapters in the background.

ation of electrical equipment already at an early age, even though my family did not do any DIY repairs.

The first device I serviced was a VHS recorder. The rubber surfaces of one of its pulleys had dried up and could no longer transfer power. Since the part in question was not available over the counter, I had to order one from a service workshop.

As my next project, I replaced a failed rectifier transformer on a television. My bag of tricks started growing with the help of some professional tips. Soon afterwards, my friends heard that I offered free repair service and started bringing me their broken bicycles and Minidisc players. Little by little, I started putting together computers, bicycles and cameras. Practising with hardware salvaged from the dumpster is a good starting point, since failures are not costly.

Belts are the most common mechanical parts that fail. You can find them in old analogue consumer entertainment devices, such as cassette and record players. Belts are also used in washing machines and cars. In four-stroke engines, for example, the valve timing is managed by the camshaft. It needs to be synchronised with the crankshaft and, possibly, other auxiliary equipment such as the oil pump. The belts will unavoidably loosen and replacing them is a regular job for service workshops.

Planned obsolescence

The international company DuPont very nearly went bankrupt when it introduced Nylon, one of its best-selling products. Nylon tights were advertised as durable, and their sales were strong. However, sales were suffering due to a lack of continuity. DuPont solved the problem by making the material weaker. This created an entirely new industrial discipline: planned obsolescence and lifetime estimates.

This practice is often justified by stating that the funds received from the sales of new products are spent on further development. The fact is, however, that international investment bankers receive the largest share of the profits. A few decades into planned obsolescence, we now face a mountain of computers, cameras, digital receivers and routers that are only a couple of years old, as well as LCD screens that are "too small".

MacBook Pro version 1.1 from 2006 and version 2.2. The latter has a 3-gigabyte memory limit.

The environmental effects of electronic waste

Industrialisation and the birth of mass production gave us a vast amount of wonderful items. However, the manufacturing processes behind them will continue to affect the environment for much longer than the service life of the items. New manufacturing methods have reduced the need for raw materials and the environmental burden caused by manufacturing, but at the same time, the service life of home appliances, for example, has fallen dramatically.

Producing new products requires a lot of work and several different phases. It includes everything from the extraction of raw materials to the manufacturing, marketing and sales of the end product. The problems created by mass production are relatively new. Only a few decades ago, most household waste was non-toxic and bio-degradable. At present, future generations face resource scarcity and severe environmental problems as a result of our use of resources.

Waste electrical and electronic equipment contains several components that can be recycled for raw materials. Some components can even be reused, and the utilisation of electronic waste is a growing field of industry. Separating precious metals from electronic waste, for example, requires using strong chemicals and is not feasible in a domestic environment. However, on an industrial scale, it is economically and ecologically more feasible than extracting new precious metals.

Technology can be built to last, but it is rarely profitable from a financial point of view. This is an enormous challenge in the field of larger and more expensive equipment, such as cars. Manufacturing a new car creates more pollution than driving an old one until the end of its useful service life. This makes it more ecological to service your car and keep it running for as long as possible. However, our consumption preferences are largely not rational and they are easily guided by marketing.

The future

The amount of waste electronic equipment is increasing. The prices of new products have fallen and service workshops receive less work. Piles of resources are taken to the landfill. A step motor from an old scanner or printer could be used in a hobbyist-built 3D printer. A hi-fi amplifier from the 1980s is proof that components can last for a long time. It still has sufficient sound quality and output power for today's applications. Switching to a new digital standard would only require replacing a small microchip and interface, but device manufacturers prefer to market and sell an entirely new device. This uses a lot of raw materials and energy.

Going forward, we will need to rely on heavily standardised modular technology where accessories are designed in a manner where the resources needed for updating and servicing them are minimised. In the future, a new model of amplifier may be a user-installable module that includes new interfaces and codecs. New materials are in development that allow for constructing edible batteries and displays.

The future may be bright, but it requires that we change our attitude in terms of waste electronic equipment. We must not dump our equipment on third world countries after a short period of use. Waste electronic equipment is a source of resources that must be utilised. 3D printing and milling offer enormous potential for manufacturing spare parts. I recommend that you learn to fix, mod and hack your own devices!

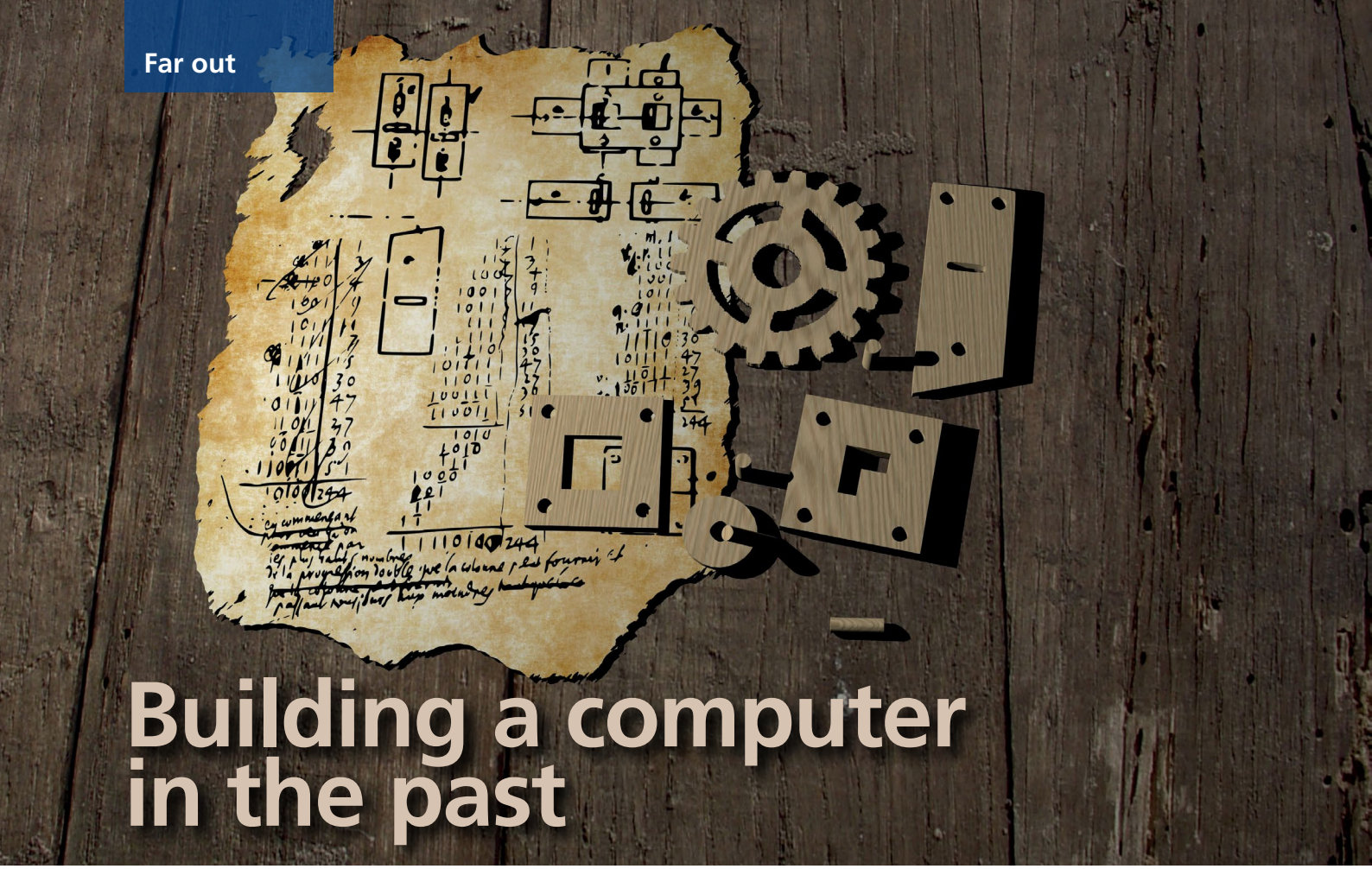
Digging up dinosaurs

Story by Mikko Heinonen

As Albert writes above, you can find usable and fairly modern PC hardware among waste electronic equipment. But there are more treasures to be found, and some of them are even classic: I have salvaged several old game consoles, processors, memory chips and even classic home computers, such as an Apple II.

Computers from the MS-DOS era are a particularly endangered species, since they are no longer in active use but have not yet been revived by the retro computing community. Nevertheless, they can still offer lots of fun. For example, you can create a nice retro workstation by replacing the hard drive with a CF card that is too small for your photos. You can then use this PC to watch old DOS demos or even play networked Doom with your friends. The Internet is full of legal downloads of older games.

For once, you can also be really picky and only accept the best components for your setup. How about a top-of-the-line Pentium MMX with L2 cache, a motherboard loaded with RAM, a Tseng Labs graphics card and a Gravis Ultrasound? All of this could be had for a few coins from the waste container of a large company and the bargain bin at a thrift store.



Building a computer in the past

Imagine that you are thrown back in time by hundreds of years. Somehow, you manage to make your life comfortable, but it is missing a true purpose – a computer, that is. Could you build one?

Story by Ville-Matias Heikkilä

Images by Ville-Matias Heikkilä, Oona Räisänen, Nasu Viljanmaa, Wikimedia Commons (Klaus Nahr, Bruno Barral)

Modern computers are electrical devices based on microchips. It would be too easy to assume that a computer cannot be built without first developing an enormous amount of different tools and techniques for the manufacture of different electronic components. Luckily, the outlook for a time traveller is not as gloomy. You can build a programmable computer by using more coarse techniques that have been used for several millennia already.

In the 1930s, the German engineer Konrad Zuse built a programmable calculator in his parents' living room. It is currently known as the Z1 computer. It read instructions off punched tape and followed them in order to perform basic arithmetic operations on 22-bit floating point numbers. You could save numbers in the work memory, read them from user input and print them out. In other words, it met all the criteria for a computer. From a time traveller's point of view, the Z1 was interesting because it was completely mechanical: the only electrical part was a motor that rotated the mechanism.

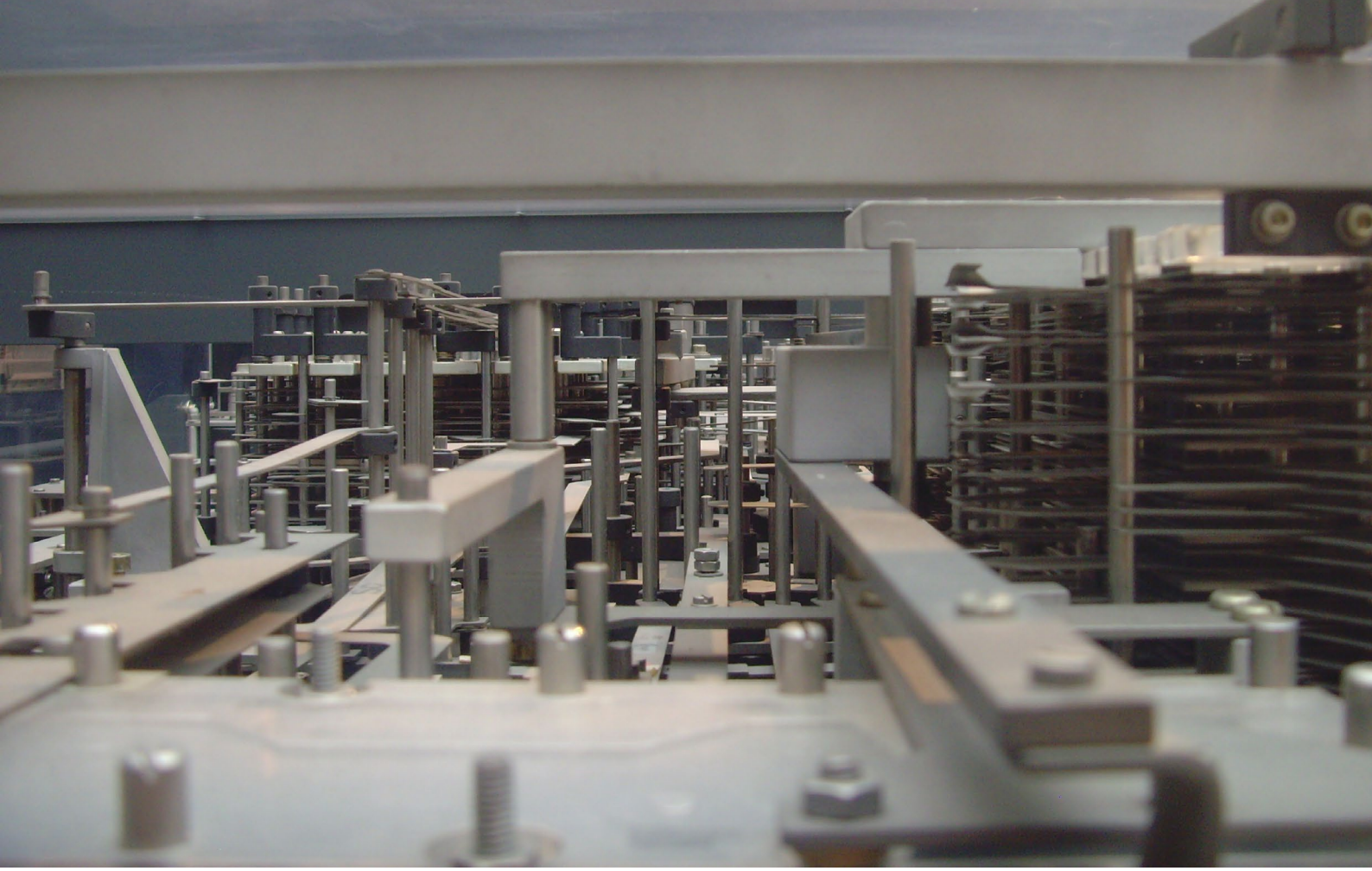
Zuse built the mechanism for the Z1 using 30,000 thin metal strips that he cut with a jigsaw. Combining strips of different sizes allowed him to build logic gates whose parts moved between two possible positions, depending on the positions of the other parts. Even the most complex binary logic is based on a few different logic gates, which means that, in principle, moving metal plates could be used to create a mechanical version of any computer.

Even the ancient Greeks were capable of constructing precision mechanics. The Antikythera mechanism that modelled the movements of the celestial bodies was based on thirty bronze gears, the largest of which had 223 teeth. The Greeks also built automatic theatres, self-opening doors, coin-operated machines, steam engines and other nearly magical devices. Hero of Alexandria, who lived during the first century, was particularly skilled at creating useless but highly innovative gadgets. It seems clear that the Greeks could have constructed the Z1 – provided that they had the blueprints. Designing the machine, however, would not have been possible without symbolic algebra and the binary system that did not arrive in Europe before the second millennium.

Gears or levers?

Binary logic built with metal strips is only one of many techniques that a time traveller can use in building their computer. Rotating gears and Leibniz wheels are also viable alternatives, since most mechanical calculators are based on them. Gottfried Leibniz developed the Leibniz wheel (stepped drum) in the 17th century for his own calculator.

The Analytical Engine, developed by the English mathematician Charles Babbage in the 1830s, would have used gears. If it had been completed, it would have been the world's first programmable computer. It could theoretically serve as a model for a time traveller's computer, as long as you did not copy its structure too exactly. The project suffered from megalomania:



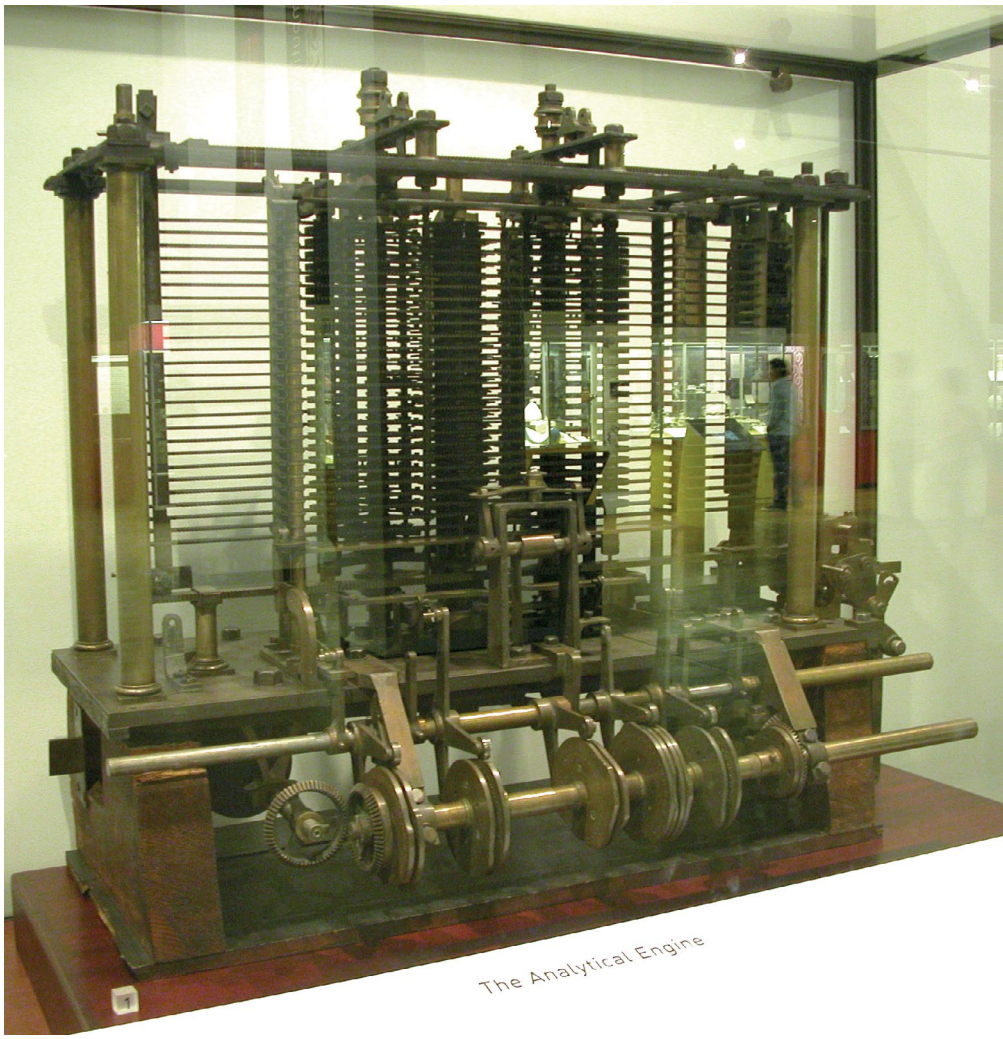
The rebuilt mechanism from the Z1. Deutsches Technikmuseum, Berlin.

the memory, for example, was supposed to contain one thousand 40-digit decimals, while Zuse made do with 64 memory locations.

Although the Analytical Engine was never completed, Babbage's humbler idea, the Difference Engine, was turned into working devices already in the 19th century. The Swede Per Georg Scheutz built a wooden prototype of his difference engine in 1843, and was later able to sell two metal versions. However, the difference engines are not programmable computers. They are only suited for producing function tables. Another predecessor of the computer from the same century was the punched card machine that was used in the US census in 1890.

Ropes, rolling marbles or pneumatics are also viable physical foundations for a computer. If the time traveller decides to invent electricity, they can build relays, electron tubes and maybe even semiconductor diodes and transistors. An optical computer, however, will most likely require a laser, so it is probably not a feasible endeavour.

Those who want to practise building a mechanical computer before travelling back in time can do so by using Lego bricks, for example. The Antikythera mechanism and Babbage's Difference



A small part of the Analytical Engine. Science Museum, London.

Engine have been built with Legos, and the British Lego hobbyist "Random Wraith" has also used them to build logic gates. But no one has yet built an entire programmable computer.

From simple to complex

Regardless of the physical and theoretical foundation of your computer, you should always build it one abstraction layer at a time. During the first stage, you develop a sufficient selection of simple and reliable primary elements, such as logic gates. In the second stage, these primary elements are used to form more complex entities, such as memory and addition elements. By increasing the level of complexity one level at a time, we will finally arrive at a machine that can run a program, after which any higher abstraction levels can be implemented in software.

The simplest logic gate is the NOT gate that takes in one bit (0 or 1) and outputs the opposite bit (1 or 0). Figure 1 explains the operation of the NOT gate on the Zuse Z1. The input bit is determined by whether the topmost plate is in the upper or lower position. The position of the left plate indicates the output bit. The right plate is used for synchronisation, without which the gate will not operate. Mechanical parts require careful synchronisation in order to work properly, so you should not try to optimise the process by removing the sync bit.

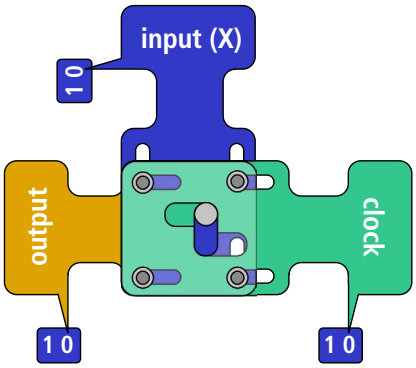


Figure 1: NOT gate on the Z1.

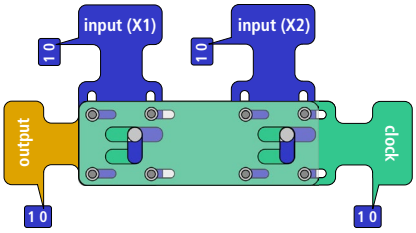


Figure 2: OR gate on the Z1.

Another gate used by Zuse was OR (Figure 2) that takes in two bits. It outputs zero if both input bits were zero, otherwise, it outputs one. In principle, binary logic can be built with only one gate type (NAND or NOR), but you can reduce the size by adding others. The XOR gate is very useful for addition, for example.

To add together two arbitrary binary numbers, we need an element known as an adder (Figure 3) that takes in three bits (A, B and C) and outputs their sum in two bits (D and E). The upper bit of the sum (D) can be routed to the input of another adder, which allows eight parallel adders to be chained together in order to form an addition circuit for two 8-bit numbers (Figure 4).

A long adder chain is not absolutely necessary, however. For example, the first Finnish computer ESKO performed additions one bit at a time, using a single adder. The smallest and slowest model of the PDP-8 minicomputer also did this. However, Zuse was not as frugal even with the Z1.

Instead of moving plates, you can also use rotating shafts and gears for the basic structure. This may even be a better option in some cases. According to Random Wraith's observations, the preferred elementary operations for rotation-based logic are the sum, difference, halving and absolute value of the number of rotations. These analogue operations can be fairly simply used to build digital circuits – both logic gates and binary adders.

Memory and buffers

In addition to the calculation elements, a working computer requires storage space for the results. The Z1 had two registers for this purpose, the R1 and R2, and RAM memory with 64 locations. Calculations were always performed between two registers and the result was saved in either of them. There were dedicated instructions for handling memory. They copied data from the registers into the memory and back.

In the Z1's RAM memory, each bit corresponded to a small metal strip whose position was altered. The strips were placed in a grid that was surrounded by the selector mechanism. The selector's first three input bits se-

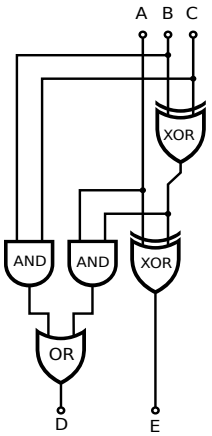


Figure 3: An adder that calculates the sum of two binary numbers.

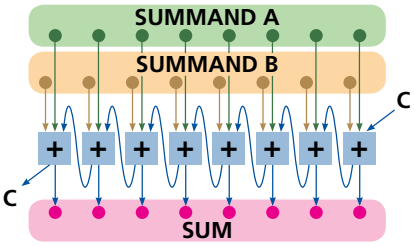


Figure 4: An adder chain that calculates the sum of two 8-bit numbers.

lected one of the eight rows on a level and the following three bits selected one of the eight columns. This allowed reads and writes to be targeted at one strip at a time. The memory was constructed by using 22 of these bit levels that were operated simultaneously.

Modern computers run their software from RAM, but the Z1 read its instructions off of punched tape, which was a fairly popular means of storage in early computing. However, punched tape and punched cards have been used for controlling different mechanical devices since the 18th century. Barrel organs and music boxes also use a type of mechanical drum memory. It could be used in computing to save a microprogram that performs the computer's actual instruction set.

Instruction set

The Z1's instruction set included arithmetic instructions, memory handling instructions and number input and output commands. The problem was the lack of jump instructions. Loops had to be performed by either writing them out or by taping the beginning and end of the tape together. Therefore, you should not model the instruction set on your computer after the Z1, since there are more advanced options.

In principle, a general-purpose computer can be astonishingly simple. Many cellular automatons are Turing complete, for example. In theory, a device that winds a memory tape around endlessly and changes the state of each memory location based on the states of the previous cells could be considered a computer. In practice, and especially in mechanical form, such a machine would be extremely slow and laborious and would only provide theoretical pleasure to even the most hardened idealist.

Many hobbyists who have built computers from their elementary components have used the instruction set of the PDP-8 minicomputer due to its simplicity. This could make it a good starting point for a mechanical computer. Other ideals could be the Data General Nova, TMS1000, RCA 1802 and MOS 6502. Those who desire a higher level of elegance might develop a Forth-style stack-based instruction set.

Should I invent electricity first?

Mechanical computers have their limitations. Even if you could make one completely reliable, it will be hopelessly slow for many interesting tasks. The fastest electromechanical computers have reached approximately ten additions per second, which would also be the upper limit for fully mechanical ones. Even using electron tubes can increase the performance by a factor of several thousand.

Inventing electricity may also prove otherwise beneficial for a time traveller. Electricity allows for building a telegraph and a radio, which may give warmongering ancient rulers the upper hand against their neighbours. Once the time traveller has sold enough communications technology to be regarded with favour by the rulers, they will also be in an excellent position to start constructing a computer.

The first electrical device that a time traveller will want to build is the battery. Chances are that they are carrying a mobile phone, and its backlight alone will make it an object of immense magical power in the eyes of the people of ancient times. Charging it is recommended in order to prolong the benefits of the magic power. The electrolyte can be sauerkraut, rowan berries, citrus fruit or vinegar, for example. You also need two different metals for the anode and cathode. Recruiting a skilled blacksmith for building the conductors and connectors is recommended.

The skills required for constructing generators, transformers and relays have been around for centuries, but our understanding of physics was insufficient and did not allow for their invention before the 19th century. An electron tube can be built with traditional artisan skills as regards the glass, metal and insulation, but pulling a vacuum inside the tube may prove to be a problem. A crude form of a vacuum pump was already known in ancient times, but a time traveller who wants to build tube computers will most likely need to develop a better method for this. Those interested in semiconductors will most likely need to spend decades developing the necessary processes before the construction of the computer can begin.

How to justify my creation?

It appears that a time traveller with sufficient skill and good fortune could build a computer in the Middle Ages or even earlier. But how would the rest of the world view such a gadget?

The boundary between technology and magic was very unclear before the Enlightenment. In ancient Greece, complex machines were mostly seen in temples that competed with each other and required a steady supply of ”miracles”. In the 16th century, the Italian scientist Giambattista della Porta wrote in his book ”Magiae Naturalis” that real magic is based on natural sciences and has nothing to do with the supernatural. In other words, a time traveller should prepare for the fact that science and technology will be categorised as witchcraft by the majority of the population.

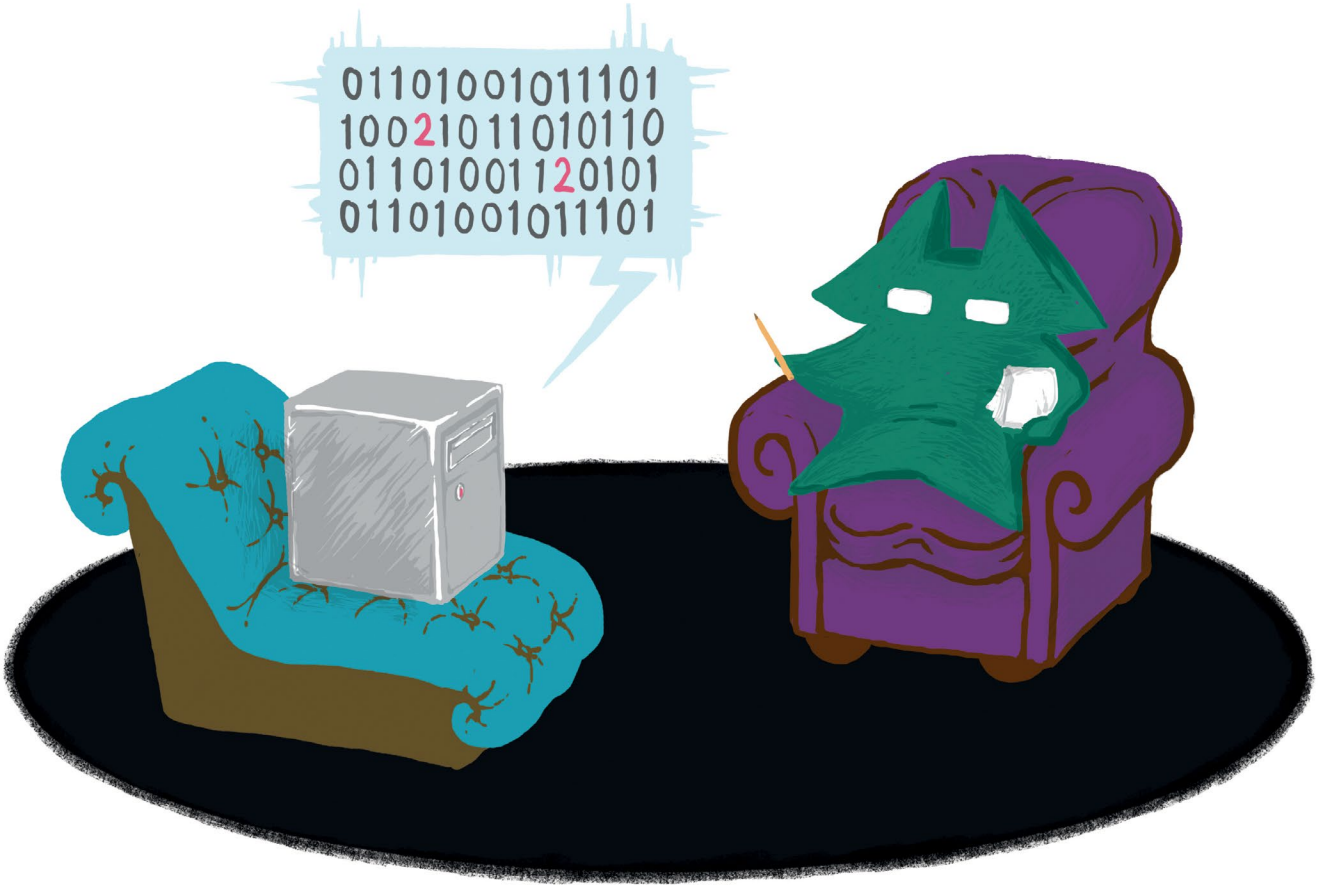
There is no point in trying to rationalise the importance of computing, either. After all, even most modern people do not understand the ideas of data processing, since computers are only collections of apps to them. The fact that the industry pioneers have struggled with finding approval for their ideas is also indicative of how difficult the concept is. After Leibniz built a working calculator, nobody resumed his work for a hundred years. Babbage's idea on the automation of brainwork was not understood, even though the automation of manual labour was already under way. Even in the 1970s, the directors of several computer companies did not believe that there would be a market for home computers.

In other words, the time traveller is not likely to encounter many people to whom they could explain the idea of a computer, even with great effort. To prevent the computer from becoming only a secret personal project, the time traveller should try to shape the culture in a more receptive direction. Becoming a philosopher might be a feasible solution. The formal systems could be injected with references to ”thinking machines” and data processing theory. If the era is very narrow-minded, however, the most radical thoughts should only be expressed at meetings with secret societies.

If the time traveller is, instead, thrown into the future, where humanity has suffered a major technological setback, this scenario has a major benefit. If people still exist, they will have at least some form of lore related to modern technology, which provides better prerequisites for understanding it. Therefore, a tribal warrior from the Neo Stone Age may well be more receptive to the idea of a computer than an ancient philosopher.

The history of computing is often told from an engineer's point of view: from mechanical parts to electron tubes to transistors and even denser microchips. However, our little thought exercise here shows that the development of culture has been at least as important. Humanity has had to undergo several changes in thinking before the idea of a computer could even come to pass. In the Middle Ages, a computer would have been completely anachronistic and incomprehensible – regardless of whether it had been made of wood or future components.

We can also reverse our scenario: if a time traveller from the far future arrived in our time, how would they view the possibilities offered by 2010s technology? Would they be able to use it to create something that is completely mind shattering, or would they prefer to advance, say, nanotechnology and quantum computing before building their magnificent invention?



Machine code:
The gateway to the computer's soul

Computer hobbyists have always considered machine code to be something extraordinary – after all, it is the closest a programmer can get to the actual hardware. Although machine code is no longer the gateway to programming magic, understanding it will help in comprehending technology.

Story by Ville-Matias Heikkilä

Images by Mitol Meerna, Ville Matias Heikkilä, Visual6502.org, AMD

Earlier, knowledge of machine code was an almost required skill for game and demo programmers, for example, but nowadays it is mostly generated by high-level compilers. Being able to read machine code is still useful, nevertheless. You can evaluate the work of the compiler and examine and modify programs without their source code. Possessing this skill makes the computer and its software much more tangible. Machine code is still an important tool for people working with vintage hardware, microcontrollers and low-level security vulnerabilities.

Machines speak many languages

Not all machines can understand the same machine code. PC processors, for example, use x86 machine code and mobile devices use ARM machine code. A single machine code is also referred to as an instruction set or architecture.

For the sake of clarity, this article focuses on four instruction sets from the annals of computing history: 6502, x86, 68K and ARM. Since the design philosophies behind these instruction sets are also quite different, they will also provide an overall picture of the types of machine code that exist.

MOS Technology's 6502 is one of the most popular 8-bit processors. The 8-bit computers from Apple, Atari and Commodore and the Nintendo NES, for example, all use it or one of its clones. The 6502's traditional competitor was the Zilog Z80, based on the Intel 8080. AVR and PIC are newer 8-bit instruction sets that are mostly used in

6502
ADC #3
01101001 00000011
"Add this number to the accumulator with carry: 3!"

X86
ADD AX, 3
00000011 11000000 00000011
"Add this byte-wide number to the word-wide register AX*: 3!"
*EAX in 32- or 64-bit code

68K
ADDQ.L #3, D0
0101011001000000
"Add the small number 3 in full width to the data register 0!"

ARM
ADD R0, R0, #3
11100010100000000000000000000011
"Unconditionally add without flags!
Put the result in R0, the addends are R0 and the number 3 without extra zeros in the end."

AVR
ADIW R25:R24, 3
1001011000000011
"Add this number to the register pair R24-R25: 3!"

PDP-11
ADD #3, R0
0110010111000000 0000000000000011
"Full-width add the contents of the memory location pointed to by the program counter (3), autoincrementing the register, to the contents of the register R0!"

Instructions from different machine code dialects, broken down to bits.

The oldest parts of the register set for the current 64-bit x86 originate from the 1970s.

embedded systems.

The Intel x86 was made famous by the IBM PC compatibles. The original instruction set was 16-bit, but it has later been radically expanded and renewed – first to 32-bit for the 386 processor, then to 64-bit at the initiative of AMD. Despite the enhancements, the different historical sediments are still clearly visible in x86 machine code.

The Motorola MC68000 was used by most computers that competed with the IBM PC until the early 1990s: the Amiga, Atari ST and Macintosh as well as most UNIX workstations. It is based on the instruction sets of larger 1970s computers and is a pure CISC (Complex Instruction Set Computer) by design.

ARM is currently the most popular instruction set. It dominates the mobile platforms, in particular, but may even replace the x86. The instruction set was originally used on the Archimedes home computer, and it became popular since it offered a lot of power with a low amount of silicon. ARM is a RISC (Reduced Instruction Set Computer). Other RISCs include MIPS, SPARC, PowerPC and AVR, for example.

Following instructions

Machine code instructions are fairly dense strings of ones and zeros. The instruction presented on page 54 performs the same task in several different machine code variants. Each instruction adds the number 3 to one of the processor's internal registers, but the bit width varies, among other things: the 6502's `adc` uses 8-bit numbers, which means that the largest sum can amount to a few hundred, while the ARM can count into the billions with its 32-bit wide calculations. The number of bits in a processor or instruction set usually refers to the maximum bit width of basic calculations.

Strings of ones and zeros are difficult to read for humans. This is why people usually process machine code in symbolic form, known as assembly language. The assembly representation can also be used to guess what the instruction does even if the instruction set is not known; for example, `ad(d)` refers to addition. The same machine code may have several different assembly language syntaxes that are used by different assembly compilers or assemblers – such as the Intel and AT&T syntaxes for the x86.

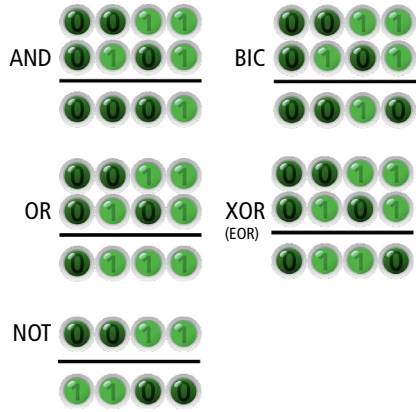
A machine code instruction usually consists of an opcode (operation code), the addressing mode and the operands. The opcode is the "verb" and it corresponds to the first word in an assembly statement, also known as a mnemonic; `add`, for example. The operands are the "nouns" that follow it: registers, numbers and memory addresses. Addressing modes can be compared to the forms of declension in human languages. They indicate how the operand part should be interpreted – whether it is a memory address or a number – and provide additional attributes; for example, the suffix `.b`, `.w` or `.l` on a 68K instruction indicates whether the operation is performed in 8, 16 or 32 bits.

Registers rotate data

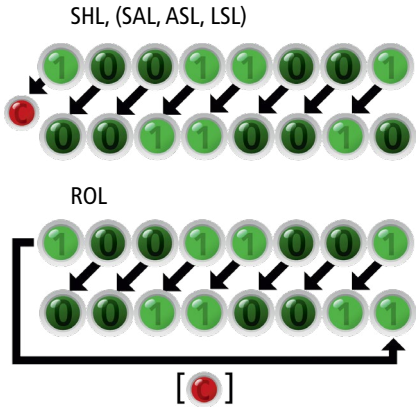
In most machine code dialects, the major part of the data processing occurs inside registers. They can be viewed as processor-internal fixed variables. The number of registers, their width and their manner of use differ substantially from one instruction set to another.

The 6502 has a very small register set and each register is tied to specific tasks. Most calculations will need to be performed in the accumulator register, A. The index registers X and Y are mostly suited for memory addressing and loop counting, which A cannot perform. In addition to these, the 6502 only has the stack pointer S, the status register P and the instruction pointer PC that indicates the memory address for the next instruction. PC is the only 16-bit register; the others are 8-bit. The limited register space is supplemented by the "zero page", the first 256 bytes of the memory, and many types of memory addressing can only be performed via the zero page.

The ARM and other RISCs, for their part, have a highly symmetrical and general-purpose register set. Theoretically, any register can be used for any purpose. The only exceptions are register R15, which is the instruction



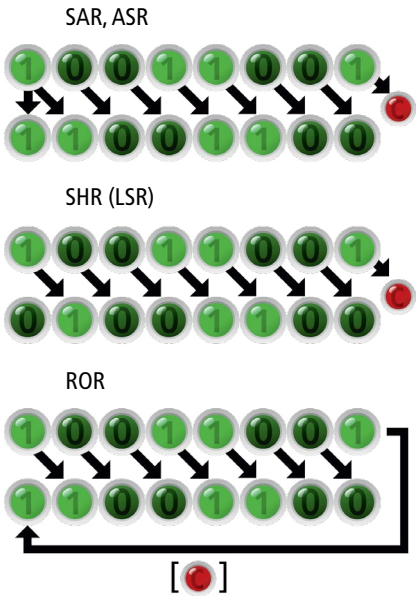
Bit operations from the instruction sets discussed in this article. BIC is used by ARM.



Operation of the bit shift instructions. Many instruction sets have different names for ROR and ROL that use the carry digit, such as RCR and RCL.

	Intel X86	68k	AT&T X86
Operand order	add destination,source	add.w source,destination	addw source,destination
Memory addressing	add ax,[1234]	add.w 1234,destination	addw 1234,%ax
Immediate	add ax,1234	add.w #1234,destination	addw $1234,%ax
Indexed address	[ebx+esi+8]	8(a0,d1.L)	8(%ebx,%esi)
Hexadecimal	1234h	$1234	0x1234
Location of the instruction	jmp $	jmp pc	jmp .
Data byte	db 123	ds.b 123	.byte 123

Assembly syntaxes are usually quite similar, but they may have some confusing differences. Here are a few examples.



8×	4×	2×	1×	Unsigned	Signed
0	0	0	0	0	+0
0	0	0	1	1	+1
0	0	1	0	2	+2
0	0	1	1	3	+3
0	1	0	0	4	+4
0	1	0	1	5	+5
0	1	1	0	6	+6
0	1	1	1	7	+7
1	0	0	0	8	-8
1	0	0	1	9	-7
1	0	1	0	10	-6
1	0	1	1	11	-5
1	1	0	0	12	-4
1	1	0	1	13	-3
1	1	1	0	14	-2
1	1	1	1	15	-1

Four-bit integers interpreted as unsigned and signed, using two's complement.

pointer, and a separate status register. The basic ARM has 16 32-bit registers, but most other RISCs have 32 or more basic registers.

The registers on the x86 were originally specialised; for example, only the registers BX, SI, DI and BP could be used for memory addressing. The 32-bit update removed some of these restrictions. Nevertheless, even the current 64-bit operation mode has some instructions that are bound to specific registers: for example, the single-byte command `stosb` saves the contents of the 8-bit AL (*accumulator low*) register to the memory location where the original DI (*destination index*) register's 64-bit extension RDI is pointing at.

The basic register set of the 68k is divided into eight data and address registers D0–D7 and A0–A7, of which A7 is used as a stack pointer. It also has a separate status register, CCR, and the instruction pointer, PC. The address registers were originally 24-bit, but they were expanded to 32 bits in the 68020. All registers can be used for calculations in a fairly general manner, but memory addressing must use the address registers.

Addressing modes modify the instructions

The simplest machine code instructions have no operands; this means that their operation is tied to specific registers. The instruction `stosb` on the x86 mentioned above is an example of this implicit form of addressing. Other examples include instructions for returning from a subroutine (`ret, rts`) and the instructions for setting and clearing flags (`sec, clc`).

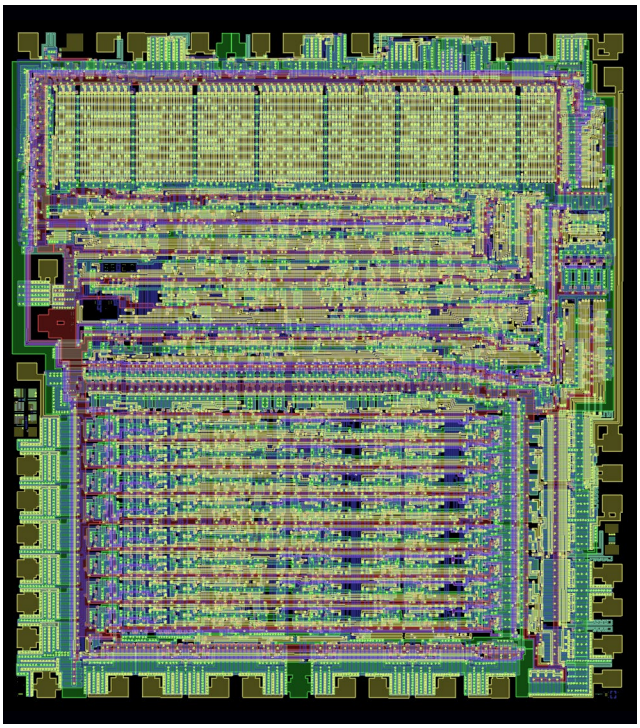
The typical number of operands in an instruction varies from one machine code to another. On the 6502, most instructions have one operand. This operand is usually a memory address, in which case the calculation occurs between the accumulator register and the memory location. The x86 and 68k have two operands: a source and

```
clc                 asl $FE
lda $FE             rol $FF
adc #$34            asl $FE
sta $FE             rol $FF
lda $FF             asl $FE
adc #$12            rol $FF
sta $FF
```

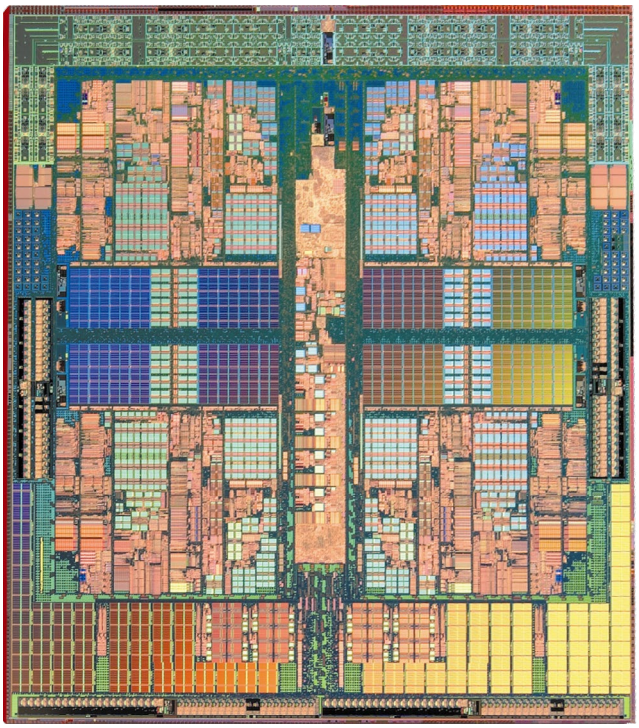
Handling 16-bit numbers with the 8-bit 6502. The example on the left adds the hexadecimal number $1234 to the value of the number saved at memory locations $FE and $FF, the one on the right multiplies it by eight by shifting the bits.

```
lp: cmp r0,r1
    subgt r0,r0,r1
    suble r1,r1,r0
    bne lp
```

A loop that calculates the largest common denominator on an ARM by using conditional execution. An Euclidean algorithm subtracts the smaller number from the larger one until the numbers are equal.



The internals of a 6502 processor. The lower half is dominated by an 8-band arithmetic and register unit, the top part has a microcode table that converts the instructions into execution steps. Between them you will find the rest of the operational logic, such as branch and flag handling.



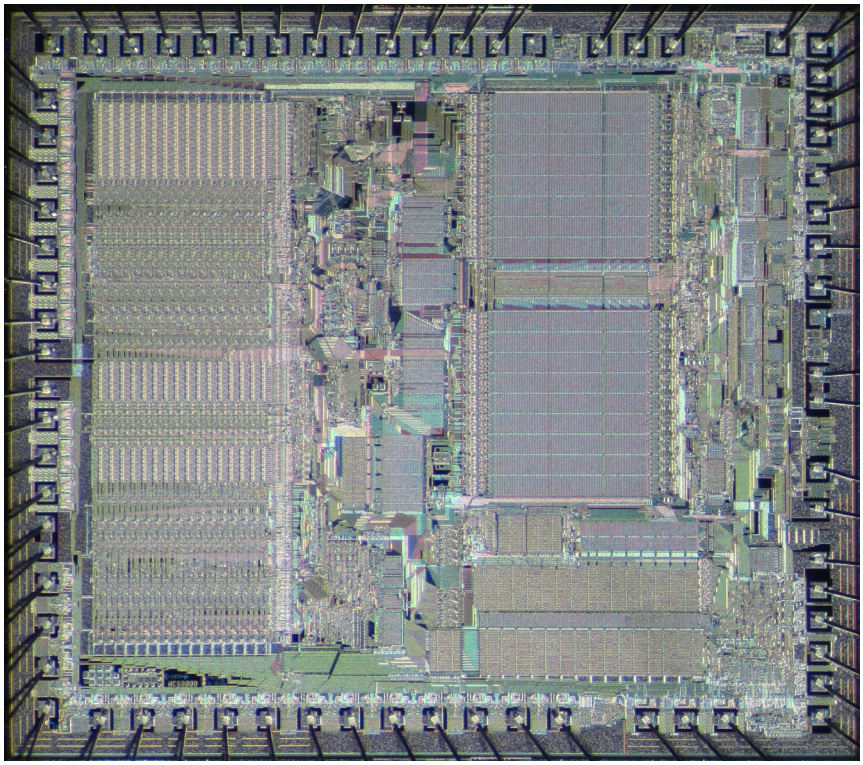
The internals of an AMD Phenom X4 processor. Most of the surface area of the four symmetrically positioned 64-bit cores is taken up by cache memory and instruction decoding and sequence logic.

destination operand for each instruction. A typical ARM instruction has three operands: two sources and one destination. Forth-style stack-based machine codes can be considered zero-operand variants.

For most processors, the main part of machine consists of operations between registers. However, *immediates* or different memory references can also be used as operands in addition to registers.

There are often limits to combining operands: on the x86, one of the operands must always be a register or an immediate; there is no direct command for "add value of memory location 2 to value of memory location 1". However, memory references can be very complex in accordance with the CISC philosophy. For example, the 32-bit x86 instruction `mov eax,[ebx+ecx*4+1256]` forms a memory address by adding together a constant and two registers, of which ECX has its bits shifted two steps to the left.

In ARM-type RISCs, most instructions can only receive registers or immediates as their operands. Memory handling must be arranged by means of dedicated load and store



The internals of a Motorola 68000. Can you find the arithmetic and register unit?

```
lp:  movem (a0)+,(d1-d7)          lp:  subcc r2,r2,#1
     movem (d1-d7),-(a1)               ldmia r0,(r3-r13)
     dbne d0,lp                        stmdb r1,(r3-r13)
                                       bne lp
```

A loop that copies the contents of a memory area in reverse order to another memory area by using the register set instructions. 68k on the left, ARM on the right.

instructions (`ld`, `st`, `mov`) that do not perform calculations.

Memory handling on the ARM and 68k is improved by addressing types where the contents of the register are incremented or decremented while the register is used for memory addressing. This is handy when scanning memory areas.

Instead of using direct addresses, it is often preferable to refer to memory locations by using the location of the instruction as a fixed point. The conditional jump instructions on the 6502 and x86 can be used to jump forward or backward by a maximum of 128 bytes; this means that the instruction only takes up two bytes. Program code that does not use direct memory addresses is called *position-independent*, since it can be executed as is from any location in memory.

Computers like to compute

Most processors use binary integers by default. The 6502, 68k and x86 also offer Binary Coded Decimals (BCD) where four bits correspond to each of the decimals 0–9. Floating point numbers, for their part, are processed with separate floating point units that have their own registers and instructions.

Negative integers are nearly always expressed as two's complements, where the sign is changed by flipping the bits around and adding one to the result. Therefore, a number that contains only ones has a value of -1, like a tape counter that goes from 000 to 999 when rewound. The same bit string can be interpreted as either signed or unsigned, and the differences become especially apparent during multiplication, division and comparison.

All machine codes offer addition and subtraction for integers (`add`, `sub`). The 8-bit machines usually lack multiplication and division (`mul`, `div`), which means that they must be implemented by means of subroutines or tables. RISCs usually only contain multiplication.

Bit operations include both logical bit operations (`and`, `or`, `eor`/`xor`) and *bit shifts* that come in many forms. The functionality of the bit operations is presented in the enclosed diagrams. The difference between an "arithmetic" and "logical" bit shifts is that in an arithmetic shift, the number is assumed to be signed and its top bit is kept in place.

EX, EXG, XCHG	exchange	Exchange the contents of the registers.
LD	load	Load from memory.
MOV, MOVE	move	Copy data from register or memory to register or memory.
POP, PL	pop, pull	Pick the topmost value in the stack.
PUSH, PH	push	Add to the top of the stack.
ST	store	Store in memory

Data transfer.

ADC, ADDX	add with carry/extend	Add with carry digit.
ADD	add	Add.
DEC	decrement	Decrement by one.
DIV	divide	Divide.
INC	increment	Increment by one.
MUL	multiply	Multiply.
NEG	negate	Switch the sign.
SBB, SBC, SUBX	subtract with borrow/carry/extend	Subtract with carry digit.
SUB	subtract	Subtract.

Basic arithmetic operations.

AND	and	AND operation by bit.
ASL, SAL	arithmetic shift left	Shift bits to the left.
ASR, LSR, SHR	[arithmetic/logical] shift right	Shift bits to the right, extending the topmost bit.
EOR, XOR	exclusive or	Exclusive OR by bit.
LSL, SHL	[logical] shift left	Shift bits to the right, extending with zero.
NOT	not	Reverse the bits.
OR	or	OR operation by bit.
ROL, RL, RCL	rotate [with carry] left	Rotate bits counterclockwise [through the C flag].
ROR, RR, RCR	rotate [with carry] right	Rotate bits clockwise [through the C flag].

Bit operations.

Sometimes, the result of the operation will not fit in the destination register. For example, the sum of two 8-bit numbers has 9 bits. The topmost bit is usually recorded in the *carry flag* (C). The carry digit is used for chaining the calculations: the instructions `adc`/`addx` and `sbc`/`sbb`/`subx` are additions and substractions that consider the carry digit from the previous calculation.

What ifs

A conditional jump is the typical machine code equivalent to the `if` clause in higher-level languages. For example, the instruction `beq`, `je` or `jz` will jump to the memory address provided as the operand if the result of the previous arithmetic operation was zero. Before the jump, it is common to use a comparison instruction, `cmp`/`cp`, which performs the subtraction without saving the result. The jump instructions are usually named from the point of view of comparison; if the result of the subtraction is zero, the numbers are equal (`e`/`eq`).

The information concerning the result is usually saved in status register bits that are known as flags. The carry flag mentioned above is one of them. Conditional jump instructions examine the status of the flags and jump if a condition is met. Typical flags include:

- The zero flag (Z) that indicates whether the result of a calculation is zero.
- The sign flag (S) or negative flag (N) that corresponds to the top bit of a result that fits in a register. For negative numbers, this is 1.
- The carry flag (C) that corresponds to the bit carried over from an arithmetic operation.
- The overflow flag (O or V) is set when the extension of the result does not fit in the carry flag.

On the 6502, x86 and 68k, each calculation instruction affects the flags.

BIT, BT, BTST, TEST	bit test	Test individual bits (AND without saving the result).
CLf	clear flag	Clear a flag (e.g. C).
CMP, CP	compare	Compare (subtract without saving the result).
Scc, SETcc	set on condition	Set the value of the register to the truth value (e.g. NE).
SEf, STf	set flag	Set a flag (e.g. C).

Comparison and flags.

Bcc, Jcc	branch/jump on condition	Jump if the condition (e.g. NE) is met.
BL, BAL	branch and link	Branch to subroutine, place return address in the link register.
DBcc, LOOP	decrement and branch, loop	Decrement the value of the register and branch if the condition is met.
JMP, JP, B, BRA	jump/branch	Branch to memory address.
JSR, JR, BSR	jump/branch to subroutine	Branch to subroutine, place return address in the stack.
RET, RTS	return from subroutine	Return from the subroutine to the main routine.
SWI, INT, TRAP, BRK, SYSCALL	software interrupt, trap, break, system call	Perform a software interrupt.

Jump instructions.

HLT	halt	Halt the processor (wait for interrupt).
NOP	no operation	Do nothing.

Other instructions.

CC, NC	no/clear carry	Carry digit = 0
CS, C	carry set	Carry digit = 1
EQ, E, Z	equal/zero	Numbers equal (zero flag set)
GT, G	greater [than]	First value > second value
LT, L	less [than]	First value < second value
NE, NZ	not equal/zero	Numbers not equal (zero flag cleared)
NS, PL	no sign, plus	Result not negative (sign = 0)
S, MI	sign, minus	Result negative (sign = 1)
VC, NO	no/clear overlow	Overflow flag cleared.
VS, O	overflow set	Overflow flag set.

Conditions (as part of instructions).

On the ARM, the effect on flags is expressed for each instruction with the suffix `cc` (*condition code*). ARM does not always require conditional jumps, since the execution of any instruction can be made conditional. For example, the instruction `addeq` operates like `add`, but it is only executed if the zero flag is set.

Stacking up other stuff

A normal unconditional jump instruction may be called `jmp`, `bra` or `b`, while a subroutine jump is called `jsr`, `bsr`, `call` or `bl`. Subroutine calls store the value of the instruction pointer. This allows the execution to resume from the place where the subroutine was called. The return instruction is typically called `ret` or `rts`.

Older instruction sets typically save the return address in a memory area known as the stack. Instead, RISCs use a register that the subroutine stacks by itself if it aims to call other subroutines. The linking jump instruction for ARM is called `bl` (*branch and link*). The link register is usually R14 and the instruction pointer is R15, so the instruction for returning from the subroutine is `mov r15,r14`.

The stack stores other things in addition to return addresses. Since the subroutines use the same registers as the main program, the values of the register values will commonly need to be stored in the stack. Stack space can also be reserved for local variables that do not fit inside the registers. The x86 and 6502 have push and pop/pull instructions that are bound to the stack pointer, whereas the ARM and 68k use regular memory handling instructions for stack handling. The ARM and 68k also have instructions for saving or loading a desired register set at once.

Calling conventions are used to keep larger programs in check. They define how parameters and return values are relayed between the main program and subroutine, and which registers the subroutine is allowed to modify.

The world is memory

From the processor's point of view, the entire outside world consists of memory. Memory is usually divided into memory cells that are the size of an 8-bit byte and have their own numeric address.

There are two main methods for storing numbers that consist of several bytes. The 68k uses *big-endian* byte order, which means that the most significant bits are stored in the first byte. The 6502 and x86 use *little-endian* byte order and store the lower bits first. ARM can operate with either byte order; little-endian is more common, however.

In simpler devices, the physical RAM, ROM and control chips have fixed areas within the memory space. In a VIC-20 program, for example, writing to address `$900F` will always affect the colour register of the video chip. More complex hardware allows for changing the memory structure visible to the program.

If the machine has more memory than the address space can hold, such as over 64 kilobytes in a 6502 based machine, *banking* is required. Banking means selecting which parts of the total memory are visible in specific areas of the memory space. Modern operating systems modify the visible structure of the memory in order to prevent different processes from accessing unauthorised memory areas. At the same time, the code is prevented from modifying the state of the processor by switching from *supervisor mode* to *user mode* during its execution.

Virtual memory means all memory visible to the program needs to correspond to physical memory. If the address space is large enough, the program may request the operating system to extend the virtual memory to the entire contents of the hard drive, for example. When the program tries to access a memory location that is not in physical memory, this causes an exception that the operating system handles by loading the desired location from the hard drive into physical memory. From the point of view of the program, the entire contents of the drive are permanently accessible in memory.

```
        !to "skrolli.prg",cbm
        *=$0801     ; Start address of the program.

; Obligatory BASIC portion: 10 SYS2061 + final zeroes:
!byte $0b,$08,$0a,$00,$9e,$32,$30,$36,$31,0,0,0

        ldx #0      ; Set counter (X) to zero.

loop0   txa         ; Copy X to A in order to
        and #15     ; calculate X AND 15.
        tay         ; Result to Y; then fetch
        lda msg,y   ; a byte from address msg+Y.

        sta $0400,x ; Copy it to the each
        sta $0500,x ; 256-byte block of the
        sta $0600,x ; screen memory at the
        sta $0700,x ; offset X.

        inx         ; Increment X.
        bne loop0   ; Repeat until rolls back to zero.

        rts         ; Return to BASIC interpreter.

msg     !scr "read skrolli!!! "
```

A 6502 example for the Commodore 64. The PRG file generated by the ACME cross-assembler can be started directly in the VICE emulator, for example.

```
        bits 16         ; Nasm to 16-bit mode.
        org 0x100       ; COM programs start at 0x100.

        mov ax,0xb800   ; Start address of screen memory
        mov es,ax       ; .. to the segment register ES.
        xor di,di       ; Set Destination Index to zero.

        mov ah,14       ; High byte of AX is the color.

loop1   mov si,msg      ; Source Index to start of text.
        mov cx,16       ; Set loop counter to 16.

loop0   lodsb           ; AL <- [DS*16+SI], SI incs.
        stosw           ; AH*256+AL -> [ES*16+DI], DI +2.
        loop loop0      ; CX decs, repeat until 0.

        cmp di,80*25*2  ; Gone through the whole screen?
        jne loop1       ; If not, continue the loop.

        ret             ; Return to the command shell.

msg     db "Read Skrolli!!! "
```

A 16-bit x86 example for MS-DOS. NASM will compile the code and create an executable COM file.

Memory speed is not a bottleneck for 1970s processors. On the 6502, for example, memory-resident tables and unrolled loops should be used in code that is critical in terms of speed, if you can fit them in memory. For modern processors, however, a calculation needs to be really complex in order to benefit from a pre-calculated table. Internal caches and smart pipelines mean that unrolling loops is more likely to slow down the code than make it faster.

Controlling devices

Computer equipment includes auxiliary chips that have their own control registers. On the 6502, 68k and ARM, these registers are visible in the memory space. However, the x86 uses separate I/O ports that are handled with the `in` and `out` instructions.

Interruptions were designed to relieve the processor from the burden of continuously polling the states of the different devices. A device can send out an *interrupt request* (IRQ) that causes the processor to stop what it is doing and move to the interrupt handling routine. In order to manage routine tasks, most operating systems execute a timer interrupt a few dozen times per second.

In its simplest form, an interrupt is no different than a subroutine call. The start address for the subroutine is fetched from a branch table according to the interrupt type and number. In modern operating systems, the interrupt also switches the processor into supervisor mode. Only an operating system that is running in supervisor mode can access external hardware, and applications perform a *non-maskable interrupt* (NMI) when they require assistance from the operating system.

Several instructions at once

The commonly used instruction sets go back several decades, but processor operation has changed significantly during this time. Parallelism has been increased, in particular.

Traditional CISC processors run only one instruction at a time. The execution of an instruction is divided into several consecutive stages that are coded in the processor's internal microcode table. On the 6502, executing an instruction consists of 2–8 stages, whereas division on the 8086 takes up over 100 clock cycles. On these processors, a programmer can calculate the execution time for their code simply by adding together the clock cycles required for the instructions and dividing the result by the clock frequency.

One of the key ideas of RISC architectures is that the execution of simple instructions may occur in parallel. The original ARM processor on the Archimedes has a three-stage pipeline: the processor saves the result from one arithmetic operation into a register while performing the next operation and reading the following instruction from memory.

Pipeline technology means that jumps are relatively costly. Executing a jump means discarding the execution stages of the instructions that follow it. There are several ways to prevent this issue. Conditional execution, used by ARM, is one of them: omitting one or two instructions is less costly than purging the entire pipeline. *Branch prediction* is a more advanced technique; the processor tries to guess whether the jump will occur and loads instructions into the pipeline accordingly. Speculative execution, on the other hand, executes both options and discards the effects of the one that did not occur.

Many processors have several parallel pipelines, allowing them to execute consecutive instructions in real time. However, consecutive instructions commonly depend on each other's results; this means that the programmer or processor should arrange the instructions in a manner where consecutive instructions do not use the same registers. In processor automation, these techniques are referred to as *out-of-order execution* and *register renaming*.

The x86 architecture has offered its fair share of challenges for processor designers. Since the 1990s, complex x86 instructions have been broken down into RISC style microinstructions that utilise the above techniques.

```
# Define the symbol _start that points the
# linker to the beginning of execution.

.globl _start
_start:

# Initialize the loop counter.

        movq $1024,%rbp     mov r8,#1024

# Execute the system call write(1,msg,15),
# where 1 is the standard output and 15 the length.
# The write call is number 1 in 64-bit Linux
# and 4 in 32-bit.

loop0:  movq $1,%rax        mov r7,#4
        movq %rax,%rdi      mov r0,#1
        movq $msg,%rsi      adr r1,msg
        movq $15,%rdx       mov r2,#15
        syscall             swi 0

# Decrement the counter, jump if not zero.

        decq %rbp           subcc r8,r8,#1
        jnz loop0           bne loop0

# Execute the system call exit(0)

        movq $4,%rax        mov r7,#1
        xorq %rdi,%rdi      mov r0,#0
        syscall             swi 0

# The string to be written:

msg:    .string "Read Skrolli!! "
```

A Linux example that uses kernel calls for 64-bit x86 (on left) and 32-bit ARM (on right). You can compile the program on the target system by using gcc -nostdlib program.s -o program or separately by calling the as assembler and ld linker.

Special instructions for special assignments

Although the basic instruction sets can more or less do everything, they often have special extensions that speed up the performance of specific tasks.

Floating point arithmetic has been a traditional requirement for scientific calculation. The idea is that numbers are presented using the mantissa and exponent, which offers a substantially larger value range than integers. PC processors started receiving integrated floating point units in the age of the 80486, but not all game consoles and mobile devices had floating point hardware even in the 2000s.

Digital signal processors (DSPs) were used to speed up the processing of image and sound data. Even basic processors started receiving DSP-type SIMD (*single instruction, multiple data*) instructions in the 1990s. Examples of SIMD extensions include MMX and SSE for the x86 and NEON for the ARM.

True to their name, SIMD instructions use several different data elements in parallel. For example, the MMX instruction `paddb mm0,mm1` interprets the values of the 64-bit multimedia registers MM0 and MM1 as rows of separate 8-bit bytes when adding them together. There are also instructions for rearranging data elements, for example.

The registers in the SSE and NEON are 128-bit, and the elements can also be floating point numbers. SSE also supports complex floating point operations such as square roots, and it has replaced the old x87 instructions on modern x86s.

In the mobile world, in particular, the same chip may contain an enormous amount of specialised arithmetic logic. For example, the Qualcomm Snapdragon 810 contains eight 64-bit ARM processor cores, each of which has three discrete pipelines and the NEON and floating point extensions. The chip also has a 288-core graphics processing unit, a 32-bit DSP and control chips that are specific to different radio protocols. Your pocket may be performing more simultaneous calculations than an old-age supercomputer.

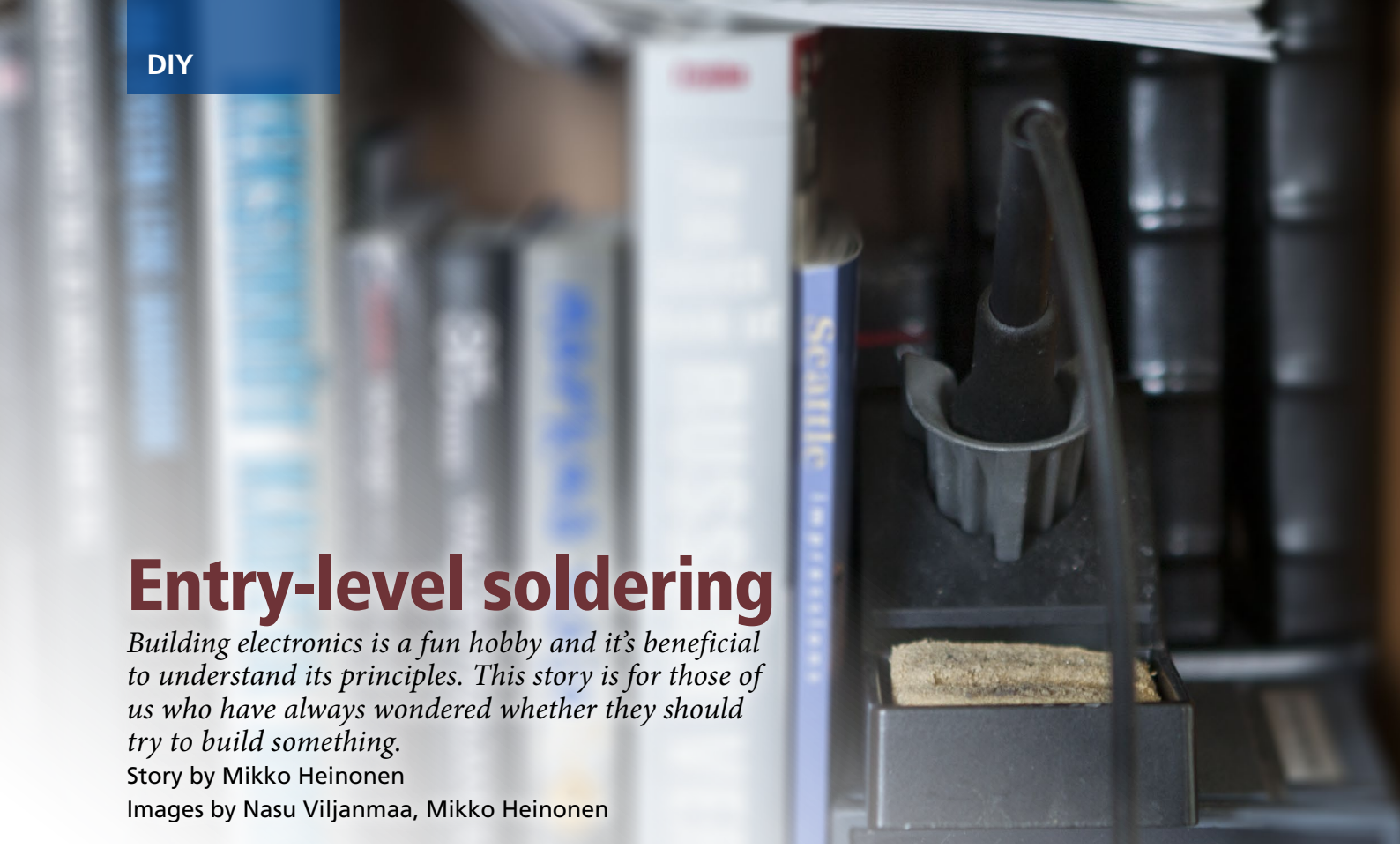
Hack away!

The most natural, and often the most rewarding, machine code projects can be found in the field of simple information technology, such as old home computers, embedded systems and electronics platforms like the Arduino. They allow for studying the operation of the device at a precision of individual bit shifts and clock cycles, and for utilising the specific features of the processor in ways that higher-level languages do not allow. *Cross-assemblers* running on a different system are typically used when writing software for these small devices, and emulators can also be leveraged for assistance. You can easily find ready-made guides for your platform of choice.

On larger computers, high-level compilers offer the easiest route to machine code; for example, using the `-S` option in GCC creates an assembly source code file that you can examine and edit. Compilers also support inline assembly i.e. embedding assembly sections into high-level language. Optimising the speed of your code is no longer a viable motivator for learning the machine code of modern languages; instead, you can use it to write programs that are as short as possible.

Other, more direct tools are also available in addition to assembly compilers. Machine code monitors and debuggers are intended for on-the-fly editing of memory and memory-resident programs. Hex editors can be used to examine and modify program files, and many of them can display an assembly representation of the file contents.

This article was a very concise look into the essence of machine code. You can use the information contained herein to examine assembly code, but you should have detailed documents concerning the instruction set and processors available before going deeper. The best way to learn the secrets of machine code is to select a suitable project and start writing code.



Entry-level soldering

Building electronics is a fun hobby and it's beneficial to understand its principles. This story is for those of us who have always wondered whether they should try to build something.

Story by Mikko Heinonen
Images by Nasu Viljanmaa, Mikko Heinonen

The last time I took a course in electronics, it was in comprehensive school over 20 years ago. I ended up on a different career path, but soldering is one of the basic skills that I have needed time after time. Knowing how to create different cables is useful, and being able to repair broken equipment can also save money.

Soldering in itself is far from rocket science: You use a soldering iron to heat up the solder until it melts and flows on the spot where you want to create a joint. Once you lift off the iron, the solder cools down and forms an electrically conductive joint between the two contact surfaces. A solder joint is more reliable than connection methods based on bare metal contact, such as compressible connectors or wire wrapping.

When working with electrical devices, it is important to stay away from anything that you don't understand. Uneducated hobbyists should keep clear of anything that involves AC power. This is why we will pick an easy project: a digital game controller. In terms of solder joints, it requires much less precision than devices using printed circuit boards.

The anatomy of a joystick

Most old home computers used the Atari pinout in their game controller ports. The name comes from the ancient Atari 2600 game console. Like the 2600 itself, the controller was very simple and consisted of only five switches: up, down, right, left and fire. Each direction corresponds to a single pin on the connector, and they also have a common ground pin. To move your character or make them fire, you simply ground the appropriate pin(s).

Digital joysticks are still generally available, but they usually only have the basic Atari functions. Since this arrangement does not use all 9 pins in the connector, a few manufacturers used the extra ones for their own purposes. For example, an MSX home computer allows using two fire buttons, and the joystick port on the Amiga is even more versatile.

In this example, I will be building an MSX joystick that has two fire buttons. Apart from this, the circuit is identical in all old home computers. It should be noted, however, that you should never connect an MSX joystick to an Amiga.

Image 1. The required tools.

Image 2. Raw materials.

Image 3. A multimeter shows that the ends of the wire are connected.

The line that goes to the second fire button in an MSX is connected to +5 volts on an Amiga; by pressing the button, you are grounding the +5 volt line, which may damage the Amiga.

You can easily get the pinout for your own device by using your search engine of choice. The query "MSX joystick pinout", for example, will provide you with a clear picture of the pin order required here. The pins are numbered from 1 (top left) to 9 (bottom right) when viewed from the front. Pay attention, as it is very easy to create a mirror image of the required connector.

Pin	MSX
1 brown	UP
2 orange	DOWN
3 grey	LEFT
4 black	RIGHT
5 red	–
6 yellow	FIRE 1
7 blue	FIRE 2
8 white	–
9 green	GROUND

Table 1. The pin arrangement of a generic extension cable.

Tools in order

You will not get very far without proper tools. At a minimum, you will require a soldering iron designed for electronics (preferably, a temperature-controlled soldering station), some solder and a pair of side-cutting pliers or a wire stripper. You also need a moist sponge or a copper cleaning pad for your soldering iron. Buying a multimeter is also a good idea, as they are very cheap and can be highly useful. A small strip of sandpaper makes prepping the soldered surfaces easier.

The soldering iron needs to be at the correct temperature. If the iron is too hot, it will melt the insulation on the wires; if it is too cold, not even the solder will melt properly. 350°C (660°F) is a good starting point. Increase the temperature when using thicker wire or thicker solder.

Image 2 shows the materials required for the controller. In theory, you could build the entire thing by using industrial microswitches, but in the interest of user comfort, I suggest you purchase an arcade stick body and two extra fire buttons of your choice.

The connecting cable will be built from an Atari joystick extension cable. There are two reasons for this. Firstly, soldering a 9-pin connector can be slightly frustrating for a beginner. Secondly, stores usually only stock these connectors with large plastic cases. On many machines, the joystick ports are located so close to each other that the connector will not fit in the port, especially when using another controller or mouse at the same time. In addition to the above, you will need a single strip of wire in order to connect the ground pin. This can usually be scavenged from a broken device you have at home.

All parts are available online. A basic arcade stick controller will cost around $10, the buttons will be $2–3 per piece and the cable will set you back around $5. You will also need to buy a case, unless you already happen to own

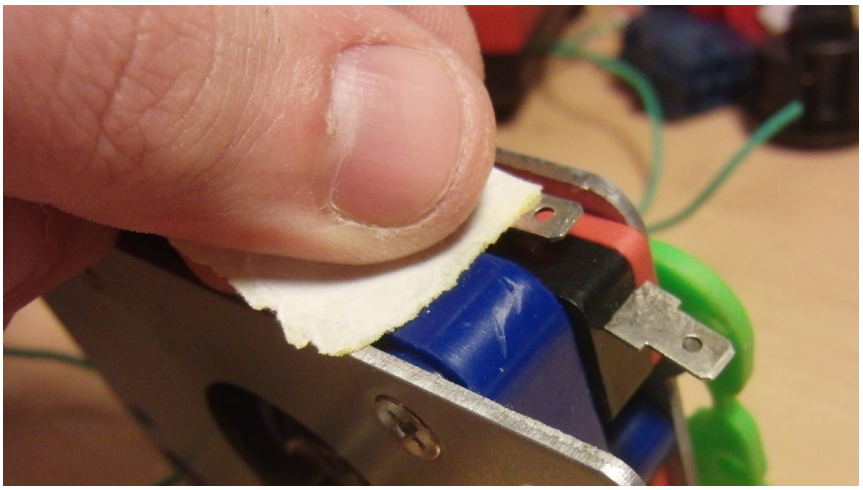


Image 4. Sand the surfaces lightly before soldering.

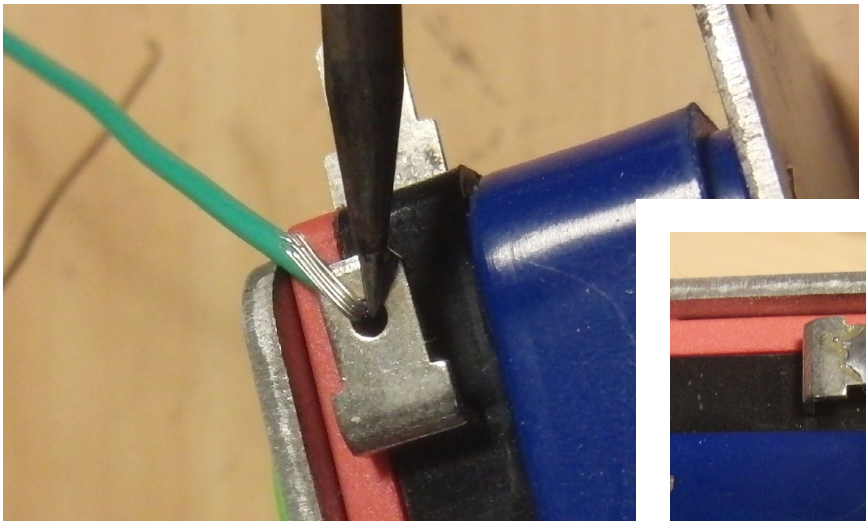


Image 5. Soldering iron in place.

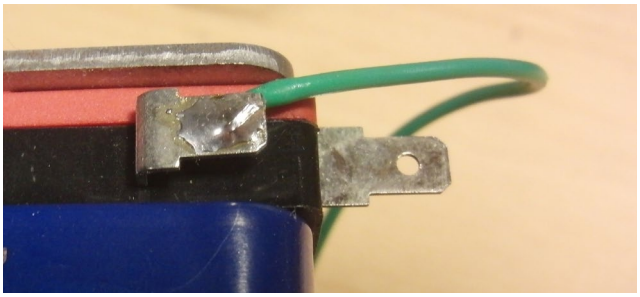


Image 6. Victory!

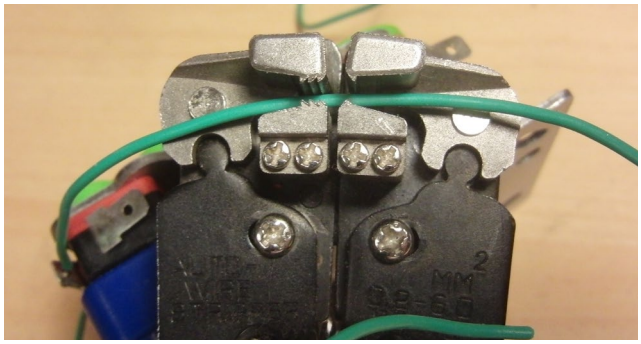


Image 7. Wire stripper.

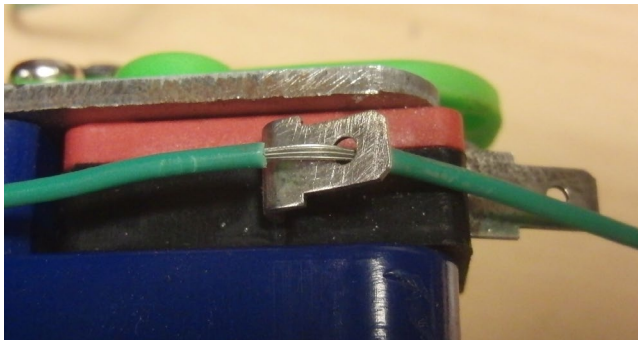


Image 8. Wire with insulation stripped in the middle.

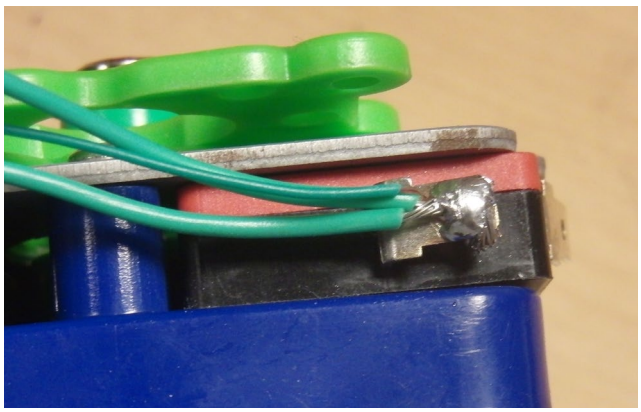


Image 9. Three wires connected.

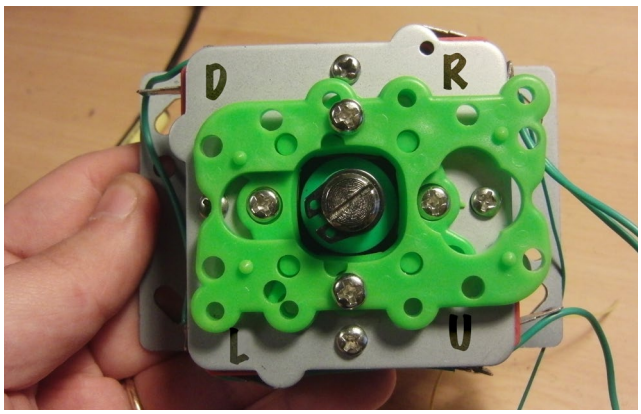


Image 10. Switch directions, viewed from below.

something suitable. For example, if you have a broken 3.5" external hard drive, you can remove and recycle the broken drive mechanism and build the joystick in the enclosure.

Get your pins straight

Since we will be cutting a few corners when building the cable, we first need to determine the pin order of the extension cable. Use side-cutting pliers or a wire stripper to cut the male end of the extension (the one with metal pins – not the one with holes). You can discard the male connector, since we will only be using the female end and the cable itself going forward. Strip the outer insulation from a length of approx. 20 cm (8 in). Then, strip the inner insulation on each wire from a length of 2.5 cm (1 in). Make sure that the wires are not touching each other.

Take out your multimeter and switch it to resistance measurement (indicated by the Omega (Ω) symbol). Wrap a small length of solder around one of the multimeter probes. Then, push the section of solder into the holes on the female end of the extension cable one at a time, going through each pin (1–9) in order. For each hole, connect the wires into the second probe one at a time until you find one that shows a reading on the multimeter (a resistance is measured, which means that current is flowing through the wire). Note the wire colour that matches each pin. Then, use the pinout of the controller you want to build to determine where the wires need to be connected. Table 1 shows an example of an extension cable bought from eBay and the MSX joystick pinout.

Before starting to solder, tie a knot in the extension cable at a distance of some 5 cm (2 in) from the point where you started to strip the outer insulation. The knot will later act as a cable clamp. Its correct location will be shown later.

Wire it up

Switch on your soldering iron and hold the stick in your hand. Since all of the switches will be connected to ground, it makes sense to connect them to each other first and then to the connecting cable. I will be using a green ground wire in the example. You can use any colour you like, as long as you remember what it is.

Take a look at your arcade stick and determine the type of switch that it uses. If the switch has only two terminals, you can connect them in any order. If there are three, they are marked NO (normally open) and NC (normally closed). For this project, we need to use the NO terminals. The NC terminals are connected when the switch is not activated, and this is the exact opposite of what we want.

The contact surface of the terminal is most likely oxidised and usually covered in protective grease. Solder will adhere very poorly to such a surface. Before attempting to solder, take a small piece of sandpaper and roughen the surface. Then, pull the wire through the hole in the terminal and position it so that it stays in place, as you will need both hands for the next step.

Press the iron against the contact surface and use your other hand to feed solder from the roll to the top of the iron. When the solder melts and attaches to the wire and roughened surface, lift off the iron. Do not keep the iron on the surface for too long in order to prevent it from heating up the inside of the switch.

If you followed the instructions above, you have now created your first solder joint. It might not be pretty, but if you have solder on the wire and the contact surface, and the wire does not come loose when you tug on it gently, then the joint is adequate for the purpose. Adding too much solder will not improve the contact – quite the opposite, in fact.

Next, go round the entire controller and solder the ground wire to one pin on each switch. You can save some additional time by using a wire stripper. Strip the wire in the middle, pull it through the hole and then solder it in place at the stripped section. This way, you will not need to cut the wire at every terminal and solder two wires at each of them.

In the fourth switch, you need to solder two other wires as well: the ground wire coming from the connector and the ground signal for the fire buttons. Strip each wire at a length of some 5 cm (2 in), wrap them together, push them through the hole in the switch and solder them in place.

Check your bearings

Next, you need to determine where to connect the signal wires. First, turn the controller so that the stick is pointing upwards. Then, decide which direction is up and mark this on the controller.

Next, turn the controller upside down and find the switch that is pressed when you push up on the stick. Do this for the other directions and mark them on the bottom of the controller. This is very important if you want your character to move correctly.

Now solder the wires for the different directions to the switch terminals. The principle is the same as for the ground wire: strip the insulation, roughen the surface, thread the wire, solder.

Encapsulation stage

Before soldering the fire buttons, you should install the controller inside a case, since most buttons are attached from below. Since my skills in plastic work are even poorer than my soldering skills, I will only be providing rudimentary instructions.

Unscrew the ball from the top of the controller, drill a 12-mm (0.5 in) hole for the shaft and insert the shaft through the hole. Reattach the ball and drill holes for the screws that fasten the body of the controller. Next, drill holes for the buttons (approx. 30 mm or 1 1/5 inch, depending on the size) and tighten them in place. Finally, create a small gap for the cable on the outer wall of the enclosure. A round file is quite useful for fine-tuning the holes – and a small electric grinder is very useful.

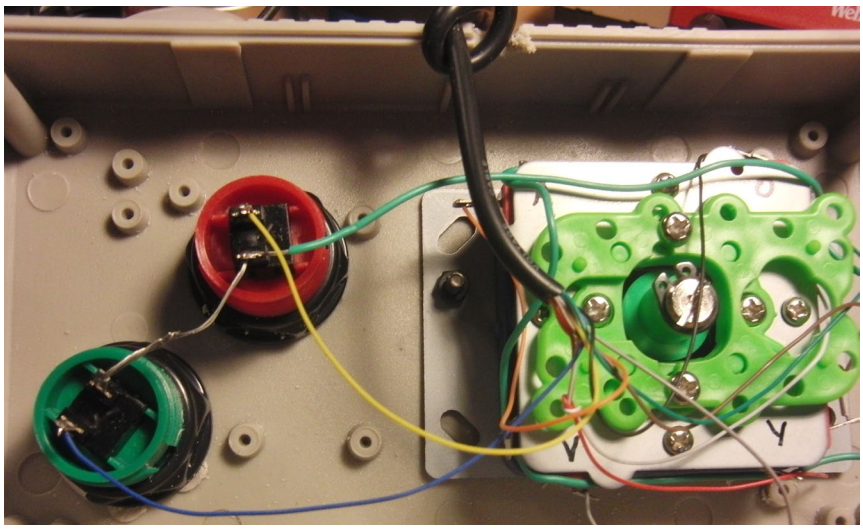


Image 11. Buttons soldered into place.

Image 12. It may be ugly, but it works.

You can also 3D print the case; Thingiverse has a few models that you can use, for example. The equipment enclosure that I am using is large, ugly and hard to work with, but I already bought one, so we will use it.

Once you have attached the controller and buttons, attach the other end of the ground wire to one of their terminals and solder it into place. Then, solder the signal wires for the buttons. In Image 11, I have stripped the insulation on the ground wire completely starting from the first button in order to make it fit through the holes. This does not matter since all the other wires are insulated.

When closing the case, make sure that the knot we made earlier is placed on the inside of the case. It will act as a cable clamp and prevent the solder joints from coming loose even if the controller is hanging by the cable.

Done!

Our functionalistic game controller box is now done. In order to ensure the electrical operation, you can use the method that I talked about at the beginning of the article – only now, you need strips of solder on both multimeter probes. Connect one probe to the ground pin and the other probe to the pin you want to test. In particular, you should make sure that you have not connected any live pins (pin 5 on the MSX, 7 on the Amiga) by mistake.

Naturally, you can improve the controller as much as you like. Better-quality sticks and buttons are widely available, and you can build the case out of wood, for example. The principle of operation remains the same. If you feel that you are not ready to solder yet, you can also use crimp connectors for attaching the wires.

At Gyro Gearloose's workshop
– hackerspaces and the thrill of making things

Hackerspaces are common spaces that are designed for building different projects. They are available to all members. What you do is not as important as your genuine motivation and interest in making things. Hackers are driven by a will to learn and do things that, perhaps, no one else has done before.

Story by Ville Ranki Images by Ville Ranki, Juuso Metsävuori

A physical space reserved for building is especially important for those living in cities where workspace for projects is difficult to come by. Machines and equipment are expensive to buy, and a hobbyist only needs most of them every now and then. Sharing the costs makes sense. Moreover, meeting like-minded people and working on projects together is fun. Currently, there are hackerspaces in at least seven cities in Finland.

Hackerspace is a difficult term. The word originates from "hack", which can be understood very broadly – it can mean writing software or creating jury rigs. The term originates from the Massachusetts Institute of Technology, and it was already used in the 1960s. Recently, the term "hacker" has been used to refer to criminals who penetrate an information network. In this article and in the hackerspace scene, the word "hacker" is used in its original meaning.

What do you want to build today?

The most common projects built in hackerspaces are related to information technology, electronics or robotics. However, calling a hackerspace a computer club, electronics club or robotics club would be an oversimplification. A hackerspace offers its members a space to build whatever they want. If the motivation is there, nothing is stopping you from cooking food or knitting a pair of socks. Indeed, most hackerspaces are expanding towards music, visual arts, biotechnology and traditional handicrafts. Examples of these include Helsinki Hacklab's courses in wearable electronics and different workshops on electronic instruments.

> *A hacker is someone who uses a coffee maker to make toast.*
> – Wau Holland, founding member of Chaos Computer Club.

Hacking projects come in many shapes and sizes. Utilitarian hacking involves repairing broken equipment or building new equipment in order to save money, such as repairing electronics or building a valve amplifier from a kit. When a product that suits your individual needs is not available, you can build it yourself – from software projects to furniture customisation. The building projects can also be related to another hobby; for example, a photographer may want to build a gyroscopic stabiliser or a rail for time lapse photography. You can also set your artistic side free by constructing a synthesizer or coding demos, for example.

The most important part is the enthusiasm for learning new things. Or course, you can buy a cheap thermometer at the supermarket, but one that uses an Arduino is a thousand times fancier and will teach you new skills. Building and experimenting is fun by itself, as long as you have the parts, tools and space available. Could I use a bucket and a smoke machine to build a smoke ring cannon like the one on YouTube?

The need for room

Of course, some projects can be built at home. Writing software only requires a computer, but when electronics or mechanical work are added to the mix, the tools no longer fit on the living room table. People living in small urban homes usually cannot set up a hacking corner in their homes; instead, working on projects requires taking out the tools before starting the work and storing them again after the work is done. This raises the threshold for starting a project, and often leads to the first step not being taken.

Moreover, mechanical work nearly always makes lots of noise. Using a circular saw or angle grinder in a flat is a sure way to get a negative reaction out of your neighbours. Sawing, drilling and laser-cutting materials create dust and smells that may also disturb them. Even a 3D printer, which seems harmless enough, will make a sound that is comparable to an old matrix printer.

The most extreme hacks involve welding, casting metals and handling corrosive chemicals. These activities require dedicated facilities and tools. The hackerspace in London, for example, has a dedicated storage for scrap bicycles and their parts as well as welding equipment that can be used to assemble new bikes from the parts.

Hackerspace vs. Hacklab

In Finland, most hackerspaces call themselves hacklabs. This is a synonym of hackerspace, like makerspace and fablab. Hacklab is easier for Finns to pronounce, but hackerspace is a more global brand. You can name your association however you like.

An extensive stock of components and parts has been found to be surprisingly useful in practice. Hacking shifts into high gear when all the parts can be found off the shelf and you no longer need to look for them in the store or order them online. Most projects are not designed that precisely in advance, and the contents of the parts shelf can offer new ideas for the implementation. Many projects start with an interesting discovery in the parts pile. Hackerspaces commonly have decommissioned equipment that, to a non-hacker, might appear to be waste. But good-quality waste electronic equipment can be a donor for displays, switches, components, motors and other useful parts. Starting a hackerspace also results in financial gain: buying tools and machines together is much cheaper.

Hackerspace Summit Finland under way in Tampere.

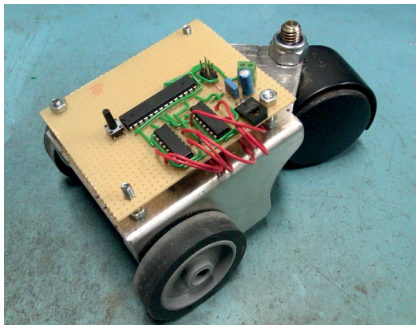
Friends and education

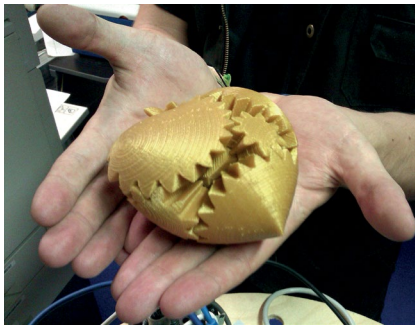
For some hackers, a hackerspace is primarily a social space for meeting friends and making things together. The same group of people commonly go out to dinner, arrange get-togethers and do other things. Hackers from different cities also cooperate; in Finland, examples of this include the Partyhat project and Hackerspace Summit Finland, which is arranged in Tampere in the winter. The different hackerspaces also have shared stands at public events. In a way, this allows hackerspaces to prevent social exclusion. At a hackerspace, no one has to pretend they are less geeky.

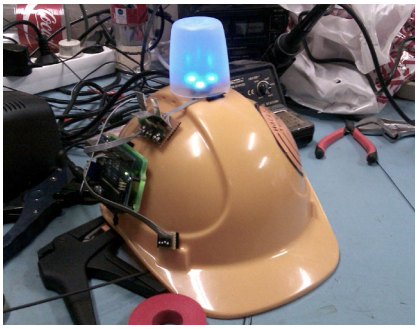
Training is an important part of the activities at a hackerspace. A typical form of this is an evening of training related to a specific topic that is held at the hackerspace. Training can also be arranged for other communities, and their members can be invited to talk about an interesting topic. Examples of training themes include the basics of electronics and programming, Arduino projects, 3D printing and using Blender 3D. Arranging training events always requires a bit of effort, but the events are always very popular. The trainer does not always need to be a specialist; basic knowledge of the topic and a willingness to share this knowledge are sufficient. The most informal training events are workshops were people work in groups and learn about a topic while the workshop leader provides assistance in case of problems.

Characteristics of a hackerspace

- **Openness**
  Anyone can join a hackerspace. Schools and companies, for example, have always had laboratories and workshops, but access to them is usually limited.
- **Independence**
  A hackerspace makes the decisions concerning its operations and is independent of external parties, such as landlords or sponsors.
- **Equality**
  Hackerspaces typically have a fairly flat hierarchy and all members are equal. The board is only responsible for the important decisions and matters related to money. Major decisions and purchases are subject to a vote. For example, members are allowed to improve the space – according to their own judgement and at their own responsibility – without asking anyone.







How to set up a hackerspace

1. Set up a core group

You will need 3 to 5 interested people and some visibility online. Create a simple website and a mailing list. In Finland, you can set up your own city.hacklab.fi domain under hacklab.fi. Add your city to the list at hackerspaces.org. Network with other hackers on IRC, by visiting other hackerspaces and by attending Hackerspace Summits.

2. Gather interested people

Arrange meet-ups at cafes, for example, and get to know each other. Advertise in places where you might find interested people. Schools, hobby clubs, workplaces and even bulletin boards at supermarkets are good places to advertise in. Gather contact details from people who want to join after you have acquired facilities for your hackerspace.

3. Set up an association and apply for financial support

In some countries, you need to be registered as an association to open an account and sign a lease agreement. It is a good idea to copy your rules from an existing hackerspace in order to ensure that they are approved. Seeking financial support should also be started at this point. Good relationships, persistence and a bit of luck will help. Note that it may take a long time to get the actual support after submitting your application. It is possible to get started without any external financial support, but it will make starting a lot easier.

4. Acquiring the facilities

Once you have enough interested people, think about the maximum sum per month that you are willing to pay for the membership. This will give you an idea of the type of lease that you can afford. Study the prices in the target area. In cities, the best locations are usually within walking distance from the centre in order to allow long-distance guests to reach the place. Contact the city, other hobbyist clubs and other similar parties to enquire about free locations. Getting a place for free or for a nominal rent would, of course, be a stroke of luck. Remember to include the cost of an Internet connection in your budget. The most important features for the facility are 24-hour access, permission to make noise and a working toilet. As the number of members increases, electronic access control will become a necessity.

5. Equipping the facilities

In the beginning, the association will probably not have any assets, and the facilities will need to be equipped with donations from the members. Used office furniture and IT equipment can be found for free, and members can volunteer their time for renovating the premises. At this point, members will usually donate their tools or materials to the association or place them on long-term loan. More expensive devices (such as 3D printers) can be financed with the income received from membership fees or by arranging a fundraiser among people interested in the purchase. Those who do not participate in the fundraiser can be requested to pay a small fee for using the device.

6. Growing your membership

Increasing the number of members is important in the beginning. Reserve a stand at every event that might attract potential members. Local events are especially important, and remember to include events that have a non-technical audience. Prepare for the fact that most people have never heard of hackerspaces and that you will need to explain it briefly and in simple terms. Invite local journalists to write about you. Arrange open nights (Tuesdays are a common choice) during which anyone can visit and learn about your activities. Arrange open nights, training and theme nights. Remember to tell your guests about what you do and involve new members in your activities from the beginning.

Larger hacker projects from Finland

- Partyhatwork – A free-form hat used at public events. It communicates with other hats over the XBee network. The hats have multicolour LEDs and they can be used to create synchronised "demo effects" over the network. Built in cooperation with several hackerspaces.
- The Chernobyl Simulator – Helsinki Hacklab's exercise in Soviet retro and multimedia. Includes a 2 m by 2 m "reactor lid" among other things.
- R100 – A remote-controlled robot built in the frame of an electric wheelchair. It can survey its environment and move autonomously with the assistance of a laser scanner.
- Ice sled – An ice vehicle that consists of a snowboard and four skis connected to a snow kite.
- Vacuum robot art project – Assisting the artist Harri Larjosto in the creation of a work where modified vacuum cleaner robots act as a moving platform.
- Metro display – Aiming to recreate Tetris on a decommissioned metro display. At the moment, printing text is already possible and the situation is developing daily.
- 24×16 LED matrix - 1.20 × 0.40 m. 16 levels of brightness per pixel, 4 NES controllers, i2c, was presented at Assembly 2012 in 8x48 format.
- "Elovalo", a work of art consisting of three LED cubes with 8×8×8 pixels in each. The work was placed in Jyväskylä's church park as part of the City of Light event in the autumn of 2012. The cubes operate autonomously with batteries and an AVR microcontroller handles the arithmetic operations.
- LazorTouch – A video wall control system built for the Vapriikki Museum Centre in Tampere. It allows the visitor to select a video clip by standing on a spot painted on the floor. Uses a laser scanner.

ILLUMINATUS

PLAYABLE DEMO ON THE COVER DISK!

The greatest game that never was

It was supposed to be the space opera of the decade – not a game, but a lifestyle. Instead, it became a meme.

Story by Janne Sirén

Back in early 1989, the Finnish computing magazine MikroBitti snatched what would become the greatest journalistic coup of its 30+ year history – an international exclusive on the mega game Illuminatus by a little known German developer.

Even in the age before the Web, this story of Illuminatus went viral across Europe, with publishing houses and press flooding MikroBitti's phone lines. Illuminatus was a game for the annals, the magazine gushed, and certainly in a way it was.

Alas, despite the great publicity, it would take 27 years before the public could play the game.

A trek to remember

The premise of Illuminatus was certainly nothing too fantastical for 1989. In that age of ever more ambitious MicroProse flight simulators, a five-disk vector graphics based space flight game for the Atari ST with a 200-hundred-page manual and a bunch of paper maps and keyboard overlays sounded par for the course. Indeed, what made Illuminatus plausible was its relative technical modesty. It would merely combine several hit game types of its era and do it well.

Like its contemporaries, Elite and Starflight, Illuminatus would use procedural content creation to generate a vast universe, far beyond the storage capacity otherwise available. Nonlinear gameplay would allow the player

to literally plot their own course. The player would start with one spaceship, a Cod Mk.I, fly around in this universe hauling cargo and, over time, accumulate money for more advanced vessels and missions.

This familiar setup already contained hints of greatness, though. Planets were not merely target destinations for those epic space treks; a Galileo MK-7 shuttle was to be taken down for flight simulation over Virus-like fractal terrains of plains, mountains and colonies. Illuminatus promised to be a great solo game.

Civilization in space

What truly set Illuminatus apart, however, was how the game was to progress. The procedurally created universe was not only immensely large, it was also alive. Artificial intelligence ran everything from space pirates and colonies to an empire's military fleet. The player's choices would affect how the universe around them changed and the universe would also evolve on its own.

As the player gained power and prominence, the game would eventually move from space and flight simulation to a tactical level and then finally to a strategic level, where the player would control entire armies and fleets, nations and worlds. Finances, diplomacy and even political assassinations would come into play.

Due to the hardware considerations of the era, the game's vantage point would, at this stage, move to that of an Empire-like strategy game. Massive surface wars and space battles consisting of thousands of ships were to be fought out between both computer opponents and up to four human players over a modem.

A star too far

That was the promise, anyway. MikroBitti interviewed the game's developers Jürgen Sternreise and Erik Dorf from Enterprise Games, who were said to have mortgaged their houses and cars to finance the game, and published a few screenshots. The article generated massive interest, with several game distributors calling MikroBitti for more information, including the British company CRL, which was impressed. Illuminatus also soon appeared for pre-order in mail-order catalogues.

However, as it happened with many ambitious mega games of the 1980s, hype got the better of Illuminatus and by autumn it was all but forgotten commercially. Persistent rumours talk of Jürgen later lamenting the lack of online crowdfunding back in the day. Thus, globally, Illuminatus never left a lasting mark, but in Finland the prominent publicity made it into a cult phenomenon.

It is not surprising then, that the abortive relaunches of Illuminatus have also started from Finland. The first resurgence of Illuminatus came in the 1990s, when the demo group Future Crew – the minds behind the likes of Futuremark and Remedy – started to work on Illuminatus for Windows. In the end, nothing came of it, either.

ILLUMINATUS

-ei enää peli, vaan elämäntapa

●● *Unohtakaa KAIKKI ennen julkaistut pelit! Illuminatus on tullut, ja ensi ajatus on: "Ei tälläistä voi ollakaan!" Lukekaa mahtavimmasta pelistä kautta aikojen ja itkekää itsenne uneen.*

"Illuminatus? Mikä himskatin Illuminatus?" Näin ajatteli allekirjoittanut raukka ja erehtyi avaamaan tanakan paketin. Kun kannessa luki "Atari ST", ja paketista löytyi liki parisataa sivuinen "Guide of the Universe", valtava tähtikartta, neljä erilaista näppäinkaaviota ja viisi diskettiä, unohtuivat työt ja juoksu kotiin alkoi. Tämä maailma häipyi, uusi astui tilalle.

Illumi-Elitekö?

Alku tuntuu tutulta. Pelaaja saa heikon avaruusaluksen ja hiukan planeetan keisariksi. Valinnanvaraa riittää, sillä maailmankaikkeus koostuu 7 miljoonasta planeetasta, ja jokaisella voi käydä. Planeetoista noin tuhannella on eriasteista elämää, ja parilla sadalla sivilisaatio.

Pelaaja aloittaa Cod Mk.I-aluksella, joka on nippa nappa rahdinkuljetukseen sopiva kevyesti aseistettu ja suojattu alus. Tarpeeksi rahaa ja rassi muuttuu Constellation-luokan taisteluristeilijäksi, Khartan-kauppa-aluksi tai muuksi noin kolmestakymmenestä vaihtoehdosta.

Laskeutumista ei suinkaan hoi- rolloimat alukset taistelevat toisiaankin vastaan, esimerkiksi kun pelaaja pakenee kahta poliisin Terminatoria ja yhtäkkiä yliavaruudesta pomppaa esiin viisi piraatti-Excaliburia. Ja muut alukset tosiaan tuntuvat älyllisen olennon ohjaamilta, varsinkin kun ne ymmärtävät ansibelilla lähetetyt "SURRENDER OR DIE" -viestit. Eipä ihme että Jurgen kehuskeli tekoälyrutiiniaan.

Niin itse alusta kuin sukkulaakin voi aseistaa kuin varustaa erilaisilla moottoreilla, aseilla, ja lisälaitteilla, joita löytyy vaatimattomat viisi ruudullista. Halvoista

kuin murto-osan verrattuna lasereihin ja keräävät itse polttoaineensa tähtienvälisistä vetypilvistä, mutta toisaalta Morita-laserit repivät aluksia kappaleiksi nopeammin kuin silmä tajuaa. Robottiohjainkortin omistajien ei tarvitse antaa kuin tulituslupa.

Ja tämä yksinäinen lentely on vasta alkua. Rahavarojen ja vaikutusvallan kasvaessa pelaaja voi palkata mukaansa palkkasotureita saattueeksi. Pahinkin piraattilaivue tai enerrealainen laivastoosasto harkitsee kaksi kertaa ennen kuin hyökkää kymmenen robottilasereilla ja partikkelitykeillä varustetun Khartan-aluksen kimppuun, joita suojelevat parhaat Hayden-generaattorit joita rahalla voi ostaa.

Pelaaja voi yrittää kasvattaa vaikutusvaltaansa kaupparuhtinaana, poliitikkona tai yksinkertaisesti avaruushunnina. Viimeksi mainittu vaihtoehto saattaa tosin päättyä ikävästi, kun Imperiumin laivasto päättää päästä keltanokasta eroon.

Keisari Kasvi

Kun valta alkaa kasvaa, ei pelaaja enää lentelekään yksin avaruudessa, vaan ohjailee kasvavaa kauppa/sotalaivastoaan mukavasti toimistostaan erinomaisella ikoniohjatulla näytöllä. Hän voi palkata salamurhaajia eliminoimaan epämiellyttäviä vastustajia, perustaa salaisia kaivoksia tuntemattomille planeetoille, kiristää kehittymättömiä aurinkokuntia Novapommeilla varoen kuitenkin Imperiumin poliisia ja salaista palvelua. Muutama lipsahdus

ja entinen huippupoliitikko l
tää lainsuojattomana etsien uu
planeettaa mistä aloittaa univ
sumin valloitus. Kannattaa siis
havarojen salliessa varustaa al
sensa kaikella mahdollisella
siirtää tarpeeksi alkupääom
Alflofolin pimeille tileille.

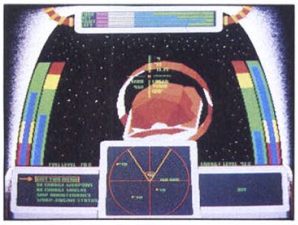
Minua viehätti suuresti m
dollisuus leikkiä toisella käde
hyväntahtoista poliitikkoa ja t
sella kädellä armotonta Maf
päällikköä. Just like real life.

Eivätkä Universumin ihme
tähän lopu. Edellä mainittuj
osioiden lisäksi peliin kuulu
niin strategiset kuin taktiset ta
telut niin planeettojen pinna
kuin avaruudessakin. Kun omi
alusten määrä kasvaa niin su
reksi, ettei ST:n kapasiteetti en
riitä sitä käsittelemään, siirrytä
taktiseen taisteluun (yleen
noin 30 taisteluun osallistuv
aluksen kohdalla) ja myöhe
min, kun puhutaan ehkä jopa t
hansista aluksista, siirrytään st
tegiseen taisteluun.

Planeettojen pinnalla käytäv
sodat muistuttavat paljon Inte
telin Empireä, mikä on suu
plussa. Planeetalle voi laskea v
miin armeijan, tai pikku armeij
la ottaa haltuun kaupunkeja ja
kaa tuottaa sotatarvikkeita.

Toteutus

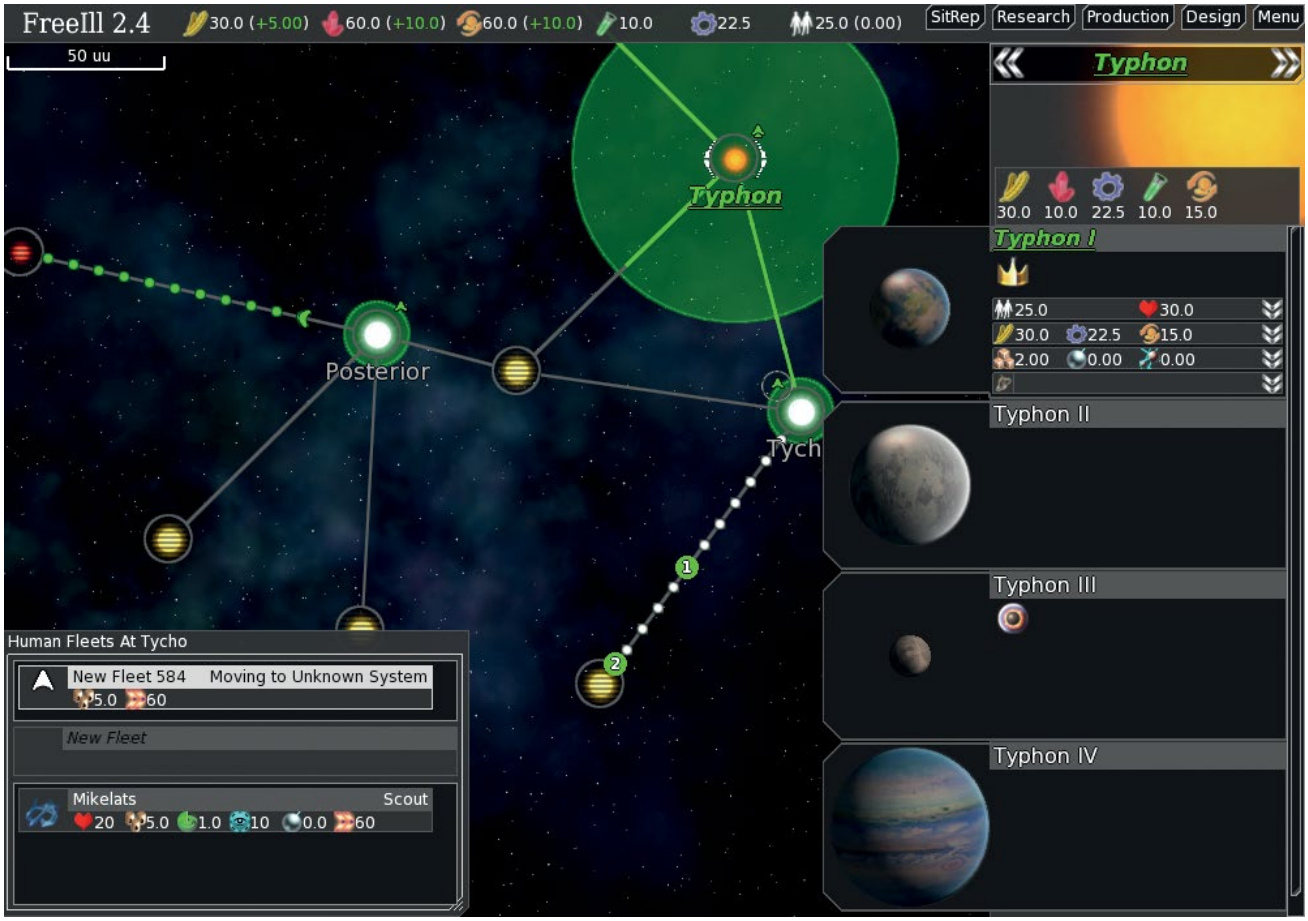
Huomaa että Illuminatuksen
hittelyyn ja ohjelmointiin on k
tetty niin aikaa kuin rakkauttak
Pelin saa siirrettyä kovalevylle
se toimii mustavalkomonitor
sakin. Nopea täytetty vektorig
fiikka sinänsä on tutun näköi
Elitestä ja muista vastaavis



Taistelun tuiskeessa.

Valmis warpp

MikroBitti published the first exclusive report on Illuminatus back in 1989.



FreeIll is one of the two Illuminatus clones whose rift Skrolli uncovered in 2014.

In a further twist, this activity reportedly spawned open-source projects. In 2014, Skrolli magazine unveiled the rift between Future Crew and community versions from FreeIlluminatus and OpenIll. According to Skrolli, Future Crew had actually abandoned their planned commercial version of Illuminatus due to death threats from one of the community projects.

We want to believe

Of course, one fact has greatly contributed to the elusiveness of Illuminatus: the whole thing was an April Fool's Day prank by MikroBitti for their 4/1989 issue. Skrolli's 1 April 2014 report on the fictitious community versions was also merely a tribute to this highly successful ruse. The prank was masterminded by MikroBitti game reviewer Niko Nirvi in 1989 and the follow-up by Skrolli's Editor-in-Chief Ville-Matias Heikkilä in 2014, both with their respective teams.

Few seemed to get the joke in 1989. The lack of mainstream Internet guaranteed several weeks' worth of attention for the hypothetical game, until MikroBitti came clean later that year. Thus, for quite a while, Illuminatus seemed to exist. The commercial interest generated back then and the subsequent Finnish cult following that has lasted to this day are very much real. Regionally, at least, Illuminatus remains a solid predecessor of Internet memes.

Riding this wave of popular sentiment created by the prank, Future Crew really attempted to put Illuminatus together in the 1990s before giving up on it, though Skrolli made up the part about the death threats. Unfortunately, neither the FreeIlluminatus nor OpenIll projects exist, but, of course, many other games have implemented and surpassed the features of Illuminatus since.

The last laugh

The butt of this joke is that, in an effort to create a video for the continuation of the Illuminatus meme for April Fool's Day 2014, Skrolli did in fact develop a private, playable PC version of a space flight scene in Illuminatus. Skrolli recreated the Illuminatus visuals, staying as true as possible to the 1989 MikroBitti screenshot mockups made originally by Petri Teittinen in Deluxe Paint II.

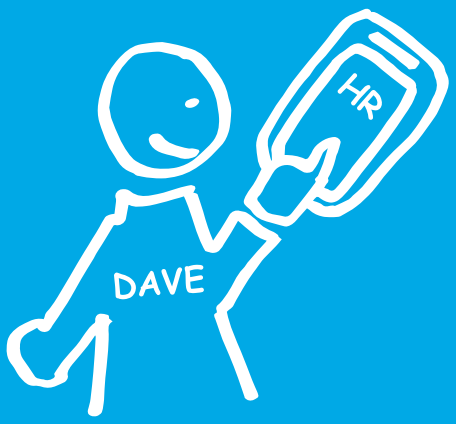
So, in the end, the prank made itself real – sort of. We have kept this version to ourselves until now, but with the launch of Skrolli International Edition in April 2016, for the very first time ever, you can find this playable demo on the virtual cover disk of this issue of Skrolli! Just scan the QR code, answer the password question and enjoy Illuminatus.

The Illuminatus of 1989 was very controversially – albeit fictitiously – made only for Atari ST. Ironically, what little existed was actually designed on an Amiga and now runs on PC. You see, platform wars, like Illuminatus, never die.

FINLAND'S LEADING IAM BOUTIQUE PRESENTS

Another day at the office.

"Well, what if I just click here? Oops..."



"OK. Terminating all employees..."

"I'm sorry, Dave. I'm afraid I can't do that."

SAVING DAVE

SINCE 2000

Spellpoint Identity & Access Management. Enabling e-services for clients. Enabling safe access for employees. Nothing but IAM. www.spellpoint.fi

Spellpoint

A. Harris, Aaron Brailsford, Aaron Puchert, Abraham Liao, Achintya Rao, Adam Blatchley Hansen, Adam Piper, Adele Asuncion, Aidan Cavanagh, Aku Kotkavuo, Alasin Media Oy, Alec Smith, Aleksey Vorona, Alessandro Cortini, Alex Hay, Alexander Calderon, Alexander Morrow, Alexander Turkovic, Alexander Wihl, Alexander Zogheb, Alexandre, Algot Johansson, Ali Kaafarani, Allen Karlowa, Amanda Werner, Ana Silva, Anders Elfgren, Andreas Flåten, Andrew, Andrew Harkness, Andrew Lee, Andrew Male, Andrew Schcik, André Koot, André Rosa, Angelo Caruso, Anssi Jaatinen, Anssi Kolehmainen, Anssi Nurmilahti, Anton Rautanen, Antti Seikkula, Antti Viklund, Ari Palo, Armando Lüscher, Armando Punzo, Arni Sumarlidason, Arttu Piironen, Ashton Davis, Asser Lähdemäki, Astro Jetson, Audun Stien, Axel Jonsson, B.A.T ry (BatMUD), Bakhshi Dashyan, Balazs Hollos, Barry White, Ben Boardman, Ben Webber, Bendert Zevenbergen, Benjamin Kittridge, Benjamin Lange, Benjamin Särkkä, Bernardas Alisauskas, Blair Harrison, Bogdan, Brad Carter, Brandon Ambrose, Brandon Bettle, Brandt Jorgensen, Brendan Grainger, Brent Rossen, Brian Murray, Bruno Antunes, Carl Angervall, Carl-Martin Pershed, Catherine Boissoneault, Charles Banas, Chris Abbott, Chris Chapman, Chris Kraft, Chris Nash, Chris Nixon, Chris Stone, Christian Rinnen, Christof Meigen, Christoph Grote, Christopher Davis, Christopher Fortin, Ciaran McElhatton, Claire Steele, Claudia Wecker, Clément Oriol, Coen Bust, Colby Fayock, Colin O'Brien, Corbin Auriti, Craig Conner, Craig Fisher, Craig McLean, Creeatist Studios, Dan Radovich, Daniel Callahan, Daniel Grankvist, Daniel Lamando, Daniel Landau, Daniel Maxwell, Daniel Milenkovic, Daniel Miller, Daniel Nitsche, Daniela Augustin, Daphne Schmidt, Dariusz Idzkowski, Darren Whiting, Dave McHugh, David Azcarraga, David Ferrandez, David Hulick, David Mery, David Pronk, David Reiss, Dean Verleger, Demir Delic, Devin Smith, Devin Valdez, Dick Steele, Drew Johnson, Drew Roberts, Duncan Sample, E. Winter, Edmund Teo, Edward Chen, Edward Thomas Burke, Eetu Korhonen, Elizabeth Cartwright, Elko Panzyk, Ellen Clark, Elvinas Ancuta, Emre, Eric Greenburg, Eric Heinrichs, Erik Crouch, Erik Rintala, Erik Schuessler, Erki Metsanurk, Eryk Sawicki, Esko Luontola, Evan Amaral, Evert van Rossum, Fabrice Charlier, Felix Penthin, Fleurot Nicolas, Frank Pietersen, Franklin Barnett, Franklin van Velthuizen, Franz Beckenbauer, Frosmo Ltd, Futurice Oy, Gabriel Fair, Gabriele D'Antona, Gareth Noyce, Garret Odegaard, Gavin Smith, Gaëtan Rivet, Gerard Ruiz Torruella, Gerrit Helling, Goutham Dodla, Grant Bowker-Bell, Grant Hutchinson, Gregory Sanders Jr., Greyson Dehn, Guilherme Gonçalves, HNF GmbH, Hannele Kormano, Hannu Viitala, Harri Granholm, Harri Koponen, Heath Manners, Hector Fiel Martin, Heesung Yang, Helmi Nykänen, Henri Nurmi, Henri Salo, Henrik Lindhe, Henryk Helsky, Hernan Silberman, Hunter Gatewood, Iikka Salmela, Ilari Heldan, Ilari Lind, Ilkka Heimo, Ilkka Pohjalainen, Ilya Margolin, Imdad Hussain, Isaac Best, Ivan Leroux, JC Ryan, Jack Reed, Jacob Wahlman, Jake, Jake C., Jake Fahrbach, James Bailey, James Klaas, James Luberda, James Luke, James Morrison, James Wright, Jamie Anker, Jani Hanka, Jani Reunanen, Jani Suhonen, Janne Heinonen, Janne Inkilä, Janne Johansson, Janne Kiiskinen, Janne Miettunen, Janne Mässbacka, Janne Peltonen, Janne Ropponen, Janne Sirén, Janne Toivola, Janne Warén, Jannik Meyer, Jared Lenahan, Jari Jaanto, Jari Tulilahti, Jari-Matti Hannula, Jari-Matti Mäkelä, Jarkko Ilomäki, Jarkko Lehti, Jarkko Sakkinen, Jarmo Kiiski, Jason McClellan, Jason Tudisco, Jasper van Eeuwijk, Jay Cornwall, Jean, Jeff Grayson, Jeff Parker, Jeffrey Guevin, Jeffrey Hofer, Jelle, Jens Füllenbach, Jeroen Zwarts, Jerome Ng, Jesper Svensson, Jesse Rosen, Jim Lott, Jimmy Devlin, Joe Dempsey, Joe Warrington, Joel Lehtelä, Joel Ohman, Johan Kiviniemi, Johannes Genberg, John Boudreaux, John Grega, John Marino, John Patterson, Johnathon Selstad, Jon Donovan, Jon Ezeiza, Jon Moroney, Jonas Wedin, Jonathan Camp, Jonathan Mukai-Heidt, Jonathan Poloff, Jonny Kelso, Joona Ruokokoski, Joonas Pihlaja, Jordan Hoff, Joris Bolsens, Jorma Oksanen, Joseph Ienna, Joseph Smyth, Joshua Case, Joshua Inatey, Joshua McCord, Joshua Melsom, Juha Klemettinen, Juha Sievi-Korte, Juha-Pekka Lautala, Juhamatti Niemelä, Jukka Hirvonen, Jules Cahon, Julien Roger, Justin Webb, Jürgen Klinsmann, Kai Nikulainen, Kalle Mansikkaniemi, Kalle Paimen, Kalle Sirkesalo, Kari Alatalo, Kari Oksanen, Kari Wassholm, Karl O'Brien, Kasper

Broegaard Simonsen, Kate Hanley, Kati Kitti, Kelvin Arcelay, Kevin Clizbe, Kevin Dixon, Kevin Jansson, Kevin Murphy, Kevin Spring, Kimmo Toivola, Klaus Bliddal, Kristoffer Just Andersen, Kyle Schreiber, L. Dodson, Lasse Immonen, Lasse Mattila, Lauri Kangas, Lauri Nurmi, Lawrence Manning, Leena Nykänen, Leif Esten Kielland, Leif Weispfennig, Liz Sippin, Loren Hersh, Lotta-Liisa Joelsson, Lukas Lozovski, Lukas Sautter, Lumi Pakkanen, Luís Alves, Maarten Jorens, Magnus Johansen, Maiju Jouppi, Maj-Britt V. Kimm, Marc Khouri, Marc Popp, Marco Cecconi, Mario Leal de Alejandro, Mariusz Marciniak, Marjo Kukkonen / iGame, Mark Mackenzie, Mark Symonds, Markku Reunanen, Marko Haarni, Marko Järvinen, Marko Milost, Markoolio Jylhä, Markus Juuti, Markus Ketzmerick, Markus Markkanen, Markus Pasula, Markus Salmijärvi, Markus Solälv, Martijn Koster, Martins Zagorskis, Mateusz Karczewski, Mathew, Mathieu Agar, Matias Mäkinen, Matias Sirén, Matt Lambie / The Frontier Group, Matt Laszewski, Matteo Sasso, Matthew Bloomfield, Matthew Gidden, Matthew Jeffries, Matthew Keehan, Matthew Key, Matthew Robeson, Matthew Vandehey, Matti Alanen, Matti Hämäläinen, Matti Tuunanen, Max Sandholm, Maxim Starodub, Meeri Kangasmäki, Meredith Lister, Michael Dankanich, Michael Falkensteiner, Michael Frnka, Michael Goubeaux, Michael Harlan Lyman, Michael Jones, Michael Kargas, Michael Kohne, Michael McLellan, Michael Nardilli, Michael Tedder, Michael Zapf, Mika Tanninen, Mikael Eriksson, Mikko Heinonen, Mikko Hypponen, Mikko Laine, Mikko Leskinen, Mikko Puustelli, Mikko Riikonen, Mikko Viitala, Moritz Bartl, Morten Sivertsen, Mounir Orfi, Mr. K. Low, Mr. Predrag Spasojevic / iGame, Nasu Viljanmaa, Nathan, Nathan Collins, Nathan Duncan, Nathan Oesterle, Neil Davidson, Neil Kelleher, Neil Lawlor, Niall Colfer, Nic Dixon, Nicholas Medau, Nick Artman, Nick Jordan, Nick Pastore, Nickolai Belakovski, Niek van de Pas, Niklas Laxström, Niko Järvinen, Niko Nirvi, Nils Sohn, Ognen Plavevski, Oleg Lavrovsky, Oleg Zhoglo, Oli Oskarsson, Oliver O'Brien, Oliver Wangler, Olivia Specht, Olli Oikarinen, Olli Pekkola, Omair Qazi, Ondrej Dolejsi, Ossi Syd, Ossi Tiltti, Owen Smith, Pami Ketolainen, Pasi Pohjanheimo, Patrick Brown, Patrik Koskinen, Paul Colea, Paul Delahunty, Paul McCulloch, Pauli Tuominen, Pavel Soriano, Pekko Lainiala, Perttu Niemi, Pete, Peter Lee, Peter Mateja, Petri Koistinen, Phil Harrison, Phil Wilkins, Philipp Nägele, Pierre Equoy, Prateek Singh, Pål Nygård, Raffael Gottardi, Raine Liukko, Ralph Brorsen, Raoul Plommer, Ray Cielencki, Reiner Herrmann, Reshef Edo, Reynard Moore, Ricardo Rebelo, Richard Prokesch, Rick Richardson, Riku Itäpuro, Riku Salkia, Riku Väkevä, Rob Rosa, Rob Salmond, Rob Uttley, Robert Vivrette, Robin van Stokrom, Romain Giot, Roman Orzegowski, Rory A. O'Connor, Ross Murray, Ross Simpson, Ryan Johnson, Ryan McBride, Sahar, Sami Kyostila, Sami Rautiainen, Sami Rosendahl, Sami Ruokoselkä, Sami Teräväinen, Sampo Töyssy / 10tons Oy, Samuel Kuehnhold, Samuel Lloyd, Samuli Koivuniemi, Samuli Siivinen, Samuli Vuorinen, Santiago Andrigo, Sean Canton, Sean Thomas, Sebastian Neuner, Sebastian Rothlauf, Sebastian Wernicke, Seth Szymanski, Silver Juurik, Simo Koivukoski, Simon Forsyth, Skyler Kehren, Sol Bekic, Sonja Dorrenboom, Spellpoint Oy, Spude, Stathis Goudoulakis, Stefan Kiel, Stefan Seemayer, Stephan Curran, Stephane Raymond, Stephen Orlando, Storme Winfield, Stuart Flawith, Sue Ilsley, Taisia Oy, Taneli Leppä, Tanya Jansson, Tapani Liukkonen, Tatu Leskinen, Taylor Breitbarth, Taylor Landicho, Ted Steiner, Teemu Likonen, Teemu Riipinen, Tero Turtiainen, Theo Holloway, Thomas Berends, Thomas Mavropoulos, Thomas Nybo Jensen, Thomas Passer Jensen, Thomas Schwery, Thor Højhus Avenstrup Jensen, Timmy Schautz, Timo Hemphill, Timo Jutila, Timo Koski, Timo Kuni, Timo Laulajainen, Timo Stordell, Timo Sulg, Tinny Nguyen, Tobias Heilmannseder, Tobias Kaptain, Tobias Preuck, Toby Butzon, Tom Jorquera, Tomer Shvueli, Tomi Liiten, Tomi Ollila, Tommi Härkönen, Tommi Tuura, Tommi Äijälä, Toni Kuokkanen, Tony Aparicio, Tony Burgio, Topher Mathrusse, Tore B. Bjoernsen, Travis Howe, Troy Martin, Troy Randle, Troy Ready, Tuomas Hokka, Tuomas Jormola, Tuomo Mäkelä, Tuomo Ryynänen, Tuomo Tammenpää, Tuuli Tammenkoski, Urs Ganse, Varun Palivela, Vesa Hirsimaa, Vesa-Matti Manninen, Vijay Raj, Viktor Elgstrand, Viktor Friberg, Ville Jouppi, Ville Ovaska, Ville Ranki, Vincent Chang, Vincent van Horssen, William Conn, William Lazaris II, Wilson, Yen Joe Tan, Ymr Stålskjegg, Youri Kersten, Yu Zhou, Zach Bryant, Zachary DiMaria, Zachary Neely, Zan

PAGES LEFT 0

ISSUE 1'20"16

CONTINUE?

YES - SUBSCRIBE
SKROLLI.FI/INTERNATIONAL